National Library of Canada

Acquisitions and Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale du Canada

Direction des acquisitions et des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

Canada

UNIVERSITY OF ALBERTA

SIMULATION-BASED PLANNING FOR CONSTRUCTION

BY

© **ANIL SAWHNEY**

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree **DOCTOR OF PHILOSOPHY IN CIVIL ENGINEERING**.

DEPARTMENT OF CIVIL ENGINEERING

EDMONTON, ALBERTA

FALL 1994

National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-315-95259-8

Canada

UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR: **Anil Sawhney**

TITLE OF THESIS: **Simulation-based Planning for Construction**

DEGREE: **Doctor of Philosophy in Civil Engineering**

YEAR THIS DEGREE GRANTED: **1994**

Permission is hereby granted to the University of Alberta Library to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves all other publication and other rights in association with the copyright in the thesis, and except as herein before provided neither the thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form without the author's prior written permission.

Anil Sawhney

Anil Sawhney
#1501, 8515, 112th Street,
Edmonton, Alberta
T6G 1K7

DATE: Aug 15, 1994

UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **SIMULATION-BASED PLANNING FOR CONSTRUCTION** submitted by **ANIL SAWHNEY** in partial fulfillment for the degree of **DOCTOR OF PHILOSOPHY IN CIVIL ENGINEERING**.

Dr. S.M. AbouRizk

Prof. S.P. Dozzi

Dr. D. W. Halpin

Prof. W. A. Weir

Dr. M. Zuo

DATE: JUNE 30, 1994

ABSTRACT

Construction projects are characterized by the random nature of the conditions under which they are implemented and by the dynamic utilization of resources. Since site conditions vary randomly and resources are dynamically utilized on various work tasks, planning of construction projects using traditional tools like critical path method (CPM) becomes limited and induces deficiency in the resulting plan. This research demonstrates that a simulation based methodology is appropriate to model such a scenario. Hierarchical Simulation Modeling (HSM) method, a simulation based method, for planning of construction projects is developed in this work. HSM utilizes the concept of "hierarchy" of work involved on a construction project. Under this concept a project is divided into operations, processes and work tasks. A resource library for the project is created after the creation of the work breakdown structure (WBS). Then based on the construction logic various operations are sequenced using serial, parallel, cyclic and hammock links. Process models are then developed using the CYCLONE method. Enhancements have been made to the CYCLONE elements to accomplish simultaneous simulation of processes using common resource pools. This includes addition of resource manipulation elements and process inter-linkage. The final step in HSM analysis is to extend all the models to a common level where simulation takes place. Planning of a bridge project (Peace River bridge) was performed using HSM to illustrate HSM concepts and its advantages. It was demonstrated that HSM will prove to be an efficient tool for project planning. The key factors for this efficiency originate from the fact that the method allows demarcation of project information into logical and manageable packages, is driven by allocation and utilization of resources, and allows advanced facilities such as cycling of operations and reusable modular model components.

Acknowledgment

The author wishes to thank his "guru" Dr. Simaan M. AbouRizk for his support, encouragement and guidance throughout the preparation of this research. Thanks to Professor Peter Dozzi and Professor Bill Weir for their encouragement, support and feedback throughout the research work. Appreciation is extended to Dr. W.J. Sproule, Dr. M. Zuo and Dr. D.W. Halpin for serving on the author's advisory committee.

The author wishes to recognize his parents for their continuous support and encouragement throughout all phases of his life.

Table of Contents

CHAPTER 1: SIMULATION OF CONSTRUCTION PROJECTS 1

1.1 INTRODUCTION 1

1.2 PROBLEM DEFINITION 3

1.3 RESEARCH OBJECTIVES 3

1.4 STATE OF THE ART 4

1.5 HIERARCHICAL SIMULATION MODELING METHOD 5

1.6 PROTOTYPE SYSTEM FOR HSM 11

1.7 THESIS ORGANIZATION 13

1.8 CONCLUDING REMARKS 13

CHAPTER 2: HSM MODELING CONCEPTS 14

2.1 INTRODUCTION 14

2.2 SPECIFICATIONS FOR PROJECT SCOPING 15

2.2.1 Work breakdown structure 15

2.2.2 Modularity concepts 19

2.2.3 Resource Definition 22

2.3 OPERATION SEQUENCING 22

2.3.1 Types of links in HSM 25

2.3.2 Rules governing the use of HSM relationships 27

2.4 PROCESS MODELS 28

2.4.1 Modeling resources at the process level 29

2.4.2 Modeling process interdependencies 31

2.4.3 Precedence rules for process modeling elements 36

2.5 CONCLUDING REMARKS 37

CHAPTER 3: COMPUTER IMPLEMENTATION OF HSM 38

3.1 INTRODUCTION 38

3.2 BACKGROUND 38

3.3 OVERVIEW OF HSM 40

3.4 IMPLEMENTATION OF HSM 41

3.4.1 Implementation of HSM modeling elements 42

3.4.2 Creation of Objects in C++ 45

3.4.3 Programming in Visual Basic with imported objects 48

3.4.4 Description of the translation module 51

3.5 SAMPLE SESSION 52

3.6 LIMITATIONS OF THE PROTOTYPE SYSTEM 59
3.7 CONCLUDING REMARKS 60

CHAPTER 4: PLANNING A BRIDGE PROJECT USING HSM 61
4.1 INTRODUCTION 61
4.2 GENERAL DESCRIPTION OF THE PEACE RIVER BRIDGE 61
4.3 SCOPING THE BRIDGE PROJECT 66
4.4 OPERATION SEQUENCING FOR THE BRIDGE PROJECT 72
4.5 PROCESS MODELS FOR THE BRIDGE PROJECT 76
4.6 SIMULATION OF THE BRIDGE PROJECT 83
4.7 EFFECT OF CHANGING THE RESOURCE ALLOCATIONS 85
4.8 EFFECT OF CHANGING THE OPERATION SEQUENCING 86
4.9 SIGNIFICANT RESULTS 88
4.10 CONCLUDING REMARKS 89

CHAPTER 5: FINAL DISCUSSION 90
5.1 SUMMARY OF THE RESEARCH 90
5.2 RESEARCH CONTRIBUTIONS 91
5.3 CONCLUDING REMARKS 92
5.4 RECOMMENDATIONS 94

BIBLIOGRAPHY 96

APPENDIX A: IMPLEMENTATION DETAILS OF HSM PROTOTYPE 101

APPENDIX B: DETAILS OF TRANSLATION MODULE 142

APPENDIX C: RESULTS OF THE CASE STUDY 164

List of Figures

Figure 1-1: Illustration of WBS and resource library in HSM 7
Figure 1-2: Illustration of operation sequencing 8
Figure 1-3: lustration of process models in HSM 11
Figure 1-4: Internal working of HSM 12
Figure 2-1: Hierarchy of work on a construction project 16
Figure 2-2: Work breakdown structure of a bridge project 18
Figure 2-3: WBS with different levels of detail 19
Figure 2-4: Modularity in Project WBS 20
Figure 2-5: Modular processes in project WBS 21
Figure 2-6: Illustration of resource library 22
Figure 2-7: Operations with common parent 24
Figure 2-8: Operation sequencing diagrams 24
Figure 2-9: Cyclic relationship 26
Figure 2-10: CYCLONE modeling elements (source Halpin, 1990) 28
Figure 2-11: HSM resource nodes 30
Figure 2-12: Illustration of resource nodes in a process model 31
Figure 2-13: General form of Class IV control structure 33
Figure 2-14: HSM process interdependency elements 33
Figure 2-15: Illustration of process inter-dependency 35
Figure 3-1: Internal structure of HSM prototype 40
Figure 3-2: Implementation strategy for the model definition module 42
Figure 3-3: Schematic representation of the modeling object 45
Figure 3-4: C++ code for object model of the QUE element 46
Figure 3-5: Object procedure for the QUE element 47
Figure 3-6: Message processing in Visual Basic 49
Figure 3-7: Visual Basic procedure for creating instances of modeling objects 49
Figure 3-8: Subroutine for "move" event for the operation object 50
Figure 3-9: Internal working of the translation module 52
Figure 3-10: Actual WBS screen 53
Figure 3-11: Input form for an operation element 54
Figure 3-12: Input form for process element 54
Figure 3-13: Resource manager screen 55
Figure 3-14: Input form for adding a resource 56
Figure 3-15: Operation sequencing screen 57
Figure 3-16: Process modeling screen 58
Figure 3-17: Input form for COMBI element 59
Figure 4-1: Schematic representation of Peace River bridge 63
Figure 4-2: Schematic representation of a typical pier 64
Figure 4-3: WBS for substructure of the bridge project 67
Figure 4-4: Modular operation for Pier-1 68
Figure 4-5: Modular operation for Pier 2, 3, 4, 5 and 6 70
Figure 4-6: Operation sequencing using the prototype 73
Figure 4-7: Operation sequencing at level-1 74

Figure 4-8: Operation sequencing for Pier-1 at level-2 74
Figure 4-9: Operation sequencing for Piers 2, 3, 4, 5 & 6 at level -2 75
Figure 4-10: Operation sequencing for Piers 2 to 6 at level-3 76
Figure 4-11: Process model for the berm process 77
Figure 4-12: Process model for the excavation process 78
Figure 4-13: Process model for the piling process 78
Figure 4-14: Process model for the cofferdam process 79
Figure 4-15: Process model for the blinding layer process 80
Figure 4-16: Process model for the formwork process 81
Figure 4-17: Process model for the rebar process 82
Figure 4-18: Process model for the concreting process 83
Figure 4-19: Revised operation sequencing 87
Figure A-1: Visual Basic Form Object 105
Figure A-2: Control Objects in Visual Basic 106
Figure A-3: Events associated with the Form Object 107
Figure A-4: Properties associated with a Control Object 108
Figure A-5: Visual Basic toolbar with HSM modeling elements 109

List of Tables

Table 1-1: Operation and process modeling elements in HSM 7
Table 1-2: Operation sequencing links 9
Table 1-3: Process modeling elements in HSM 10
Table 3-1: HSM modeling elements 43
Table 4-1: Quantity estimate for Peace River Bridge (ATU, 1989) 62
Table 4-2: Quantities for the typical bridge pier (ATU, 1989) 64
Table 4-3: Resources allocated to the bridge project 71
Table 4-4: Summary of the results for different resource allocations 85
Table 4-5: Summary of results for revised operation sequencing 88
Table A-1: Custom control files for HSM modeling elements 102
Table A-2: Custom control source code files 104
Table A-3: List of the tables in the project database 112
Table A-4: Modular operation and modular process database 113
Table C-1: Processes for the Peace River bridge project 164
Table C-2: Simulation results for the deterministic case 168
Table C-3: Simulation results for the stochastic case (mean of 30 runs) 171
Table C-4: Simulation results for the resource scenario-1 174
Table C-5: Simulation results for the revised operation sequening case 177

Chapter 1: Simulation of Construction Projects

1.1 INTRODUCTION

Implementation of large and complex construction projects requires resources such as, labour, heavy and sophisticated equipment, and materials. In order to succeed in such endeavors it is essential to deploy effective techniques to plan for resources (in this thesis the word "plan" is used in generic sense to mean a construction schedule that provides information regarding duration and sequence of work tasks and resource requirements). It is important that proper planning and control of the involved activities and resources be performed. In addition to detailed and accurate design, a detailed and accurate plan is required to achieve the project objectives. To coordinate the efforts required on the project, it is desirable to use some form of representation of the designed facility before it is actually constructed. Many techniques have been developed to assist a construction engineer in performing the functions of planning and control.

A graphical representation of the project on a time scale is considered to be practical planning tool. Initially, there was no generally accepted formal procedure for project planning. It was more of an art than a science. As the complexity of projects increased managers started using simple tools like bar charts and velocity diagrams. Then network techniques became popular in the construction industry. Network techniques such as Critical Path Method (CPM), Program Evaluation and Review Technique (PERT) and Precedence Diagramming Method (PDM) show the project activities, duration, and logical constraints as well as the start and finish times of the activities and the critical path.

Network techniques are widely used in the construction industry. Over the years it has been found that these techniques provide limited modeling versatility. The shortcomings of these methods are numerous and have been documented in many publications (e.g. McCrimmon and Rayvec, 1964 and Pritsker et. al. 1989). The deterministic and static nature of representation of these techniques is the main reason for ineffectiveness in modeling a dynamic and stochastic system such as a construction project. The following modeling assumptions make the network techniques ineffective in portraying an actual project (Pritsker et. al. 1989):

1. Branching is done on a deterministic basis.
2. No cycling or feedback is allowed in the network.
3. Projects or portions of projects are always completed successfully as the concept of failure is non-existent (CPM simply adds time at which nodes of the network are realized).
4. No explicit storage or queuing concepts are available.
5. Resource handling does not model dynamic utilization and allocation.

Computer simulation is a tool that can be used to model random phenomenon and to integrate variability into project management techniques. Van Slyke (1963) and Pritsker (1979) discussed the development of Graphical Evaluation and Review Technique (GERT) for extending the modeling capabilities of CPM and PERT. These experiments led to the use of simulation in project planning and control. Pritsker et. al. (1989) provide numerous illustrations of the application of simulation for project planning.

1.2 PROBLEM DEFINITION

Planning of construction projects using simulation overcomes the deficiencies of traditional network based techniques. Simulation allows modeling of random conditions under which projects are implemented and allows incorporation of the dynamic behavior of the involved resources. Construction simulation, especially process modeling, has matured over the years with numerous examples given in Halpin (1990), and Halpin and Riggs (1992).

The current simulation modeling methods are not geared towards combined simulation of the processes using a common resource pool, a situation that is normally encountered on a project. To simulate a construction project using the current modeling method it is essential to develop a single model that captures the details of all the underlying processes in the project. The complexity of such a model would increase as the size of the construction project increases. It is easy to envision the difficulties involved with developing a single project level simulation model. This clearly indicates that there is a need to develop a modeling framework that allows a systematic division of a construction project into a number of processes that can then be simulated simultaneously.

1.3 RESEARCH OBJECTIVES

The focus of this research was towards the development of a modeling method for the simulation of construction projects. The main objectives were:

1. Development of a hierarchical and modular modeling framework that assists in defining project level simulation models.
2. Testing of the proposed modeling method using an actual case study.

A secondary objective was the automation of the developed modeling method in a form that will allow testing and validation of the modeling framework.

1.4 STATE OF THE ART

Past studies in the area of project planning were mainly directed towards analyzing deficiencies of the existing planning techniques like CPM and PERT. Most of the previous simulation-based project planning models were hybrid in nature and generally involved a Monte-Carlo simulation of a project depicted by a network technique. A representative summary of these models is provided in the following paragraphs.

Ang et. al. (1975) developed a technique called Probabilistic Network Evaluation Technique (PNET). This technique applies probability theory to reduce the number of possible critical paths. PNET evaluates the expected project duration based on representative paths in the network. Crandall (1976) used the Monte Carlo simulation technique to carry out probabilistic scheduling. He developed a tree structure for network representation to perform the Monte Carlo experiments. Riggs (1989) summarized the attempts made to use simulation modeling for the planning of construction projects. Woolery and Crandall (1983) provided a stochastic network model for scheduling. Ahuja and Nandakumar (1984) presented a detailed approach based on simulating a project network by first specifying possible uncertainty factors and then simulating various scenarios to obtain a statistical distribution of the activity duration based on the incorporated effects.

Dabbas (1981), and Dabbas and Halpin (1982) described the development of a simulation technique that provides a planning tool for construction projects coupled with a

process level analysis. This technique provided a basic step towards applying simulation to complex and large construction projects. A project is scheduled by developing a CPM network in which some of the activities have individual CYCLONE models attached to them. The limitation of this technique is that resource allocation is done individually for each cyclic activity. In actual practice this is not the case and resources can be best modeled by common resource pools at various levels and stages of the project.

CIPROS, an object-oriented, interactive system for developing discrete-event simulation networks and simulating construction plans, was recently developed by Odeh et al. (1992). CIPROS provides a method by which modelers (in this thesis the word "modeler" has been used in generic sense to mean a construction scheduler or planner) can divide their work into various steps, but limited hierarchical modeling is achieved. In CIPROS process interactions and process interdependencies are not modeled because process models are attached to the CPM activities that are externally connected to each other.

1.5 HIERARCHICAL SIMULATION MODELING METHOD

In this research a modeling framework called Hierarchical Simulation Modeling (HSM) method was developed. This method provides a structured technique for soliciting project information to develop simulation models at a project level. This allows simulation-based planning of the project.

The plan development for a project using HSM is performed in five simple steps:

1. Develop the work breakdown structure by identifying operations and processes.

2. Develop the project resource library by defining the resources allocated to the project.
3. Sequence the identified operations using serial, cyclic, parallel and hammock links.
4. Develop process models to define the project work tasks using CYCLONE (Halpin, 1977).
5. Simulate the project, analyze the results and produce output.

The first step pertains to the division of the project into a hierarchical structure, often referred to as the work breakdown structure (WBS). In HSM the hierarchical structure of the project has to be defined such that there are at least two levels, one is the project level and the other is the process level. Therefore, for a simple project the WBS would have a project level and a process level. On the other hand a complex project can have a project level, a process level, and multiple operation levels. This feature of HSM allows the modeler to logically divide large amounts of project information into manageable components. Figure 1-1 illustrates a partial WBS for a high-rise building project. The project is first divided into "Zone-A" and "Zone-B" operations (for simplicity a partial division of only "Zone-A" is shown). Three processes, namely: "formwork", "rebar" and "concrete", are identified for the "Concreting operation" at the lowest level. Figure 1-1 also illustrates the modularity feature of HSM. Because the structural design of floors 2, 3, 4 and 5 are similar, the modeler defines a modular model for floor-2 and then refers this to the remaining floors. This feature reduces the complexity of the input required for the plan. A project WBS is developed using two modeling elements that include "operation-

element" and "process-element." Table 1-1 shows these two modeling elements, their interpretation and usage.

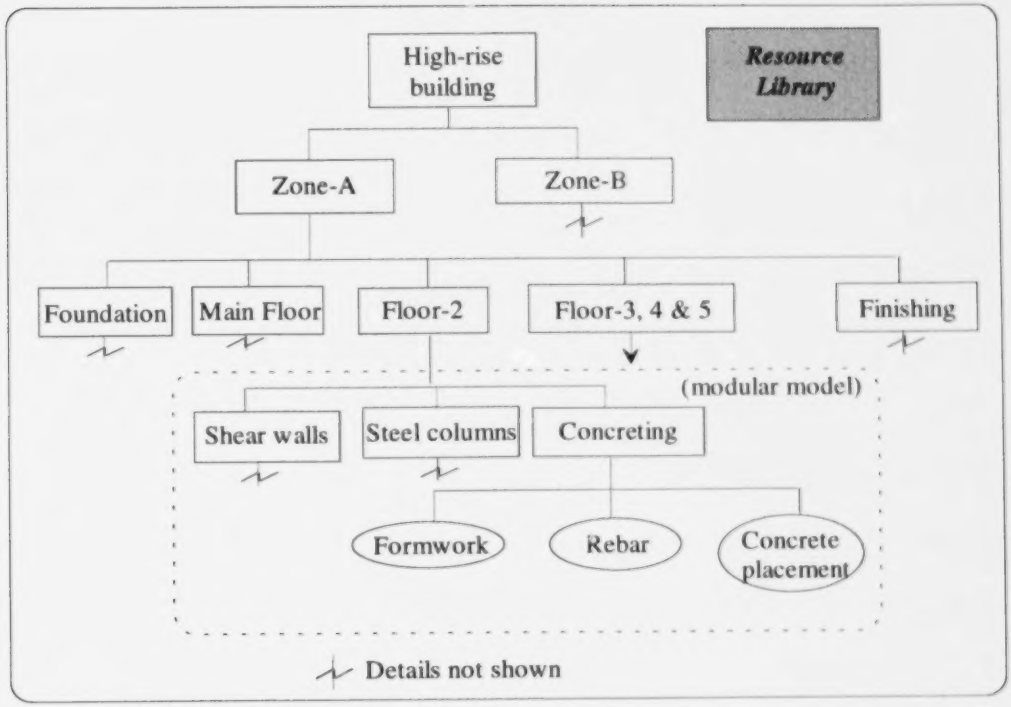


Figure 1-1 Illustration of WBS and resource library in HSM

Table 1-1: Operation and process modeling elements in HSM

Modeling Element	*Interpretation*	*Usage*
Operation	Focus on construction methods.	It can only have one parent and multiple children in the hierarchy. The parent element can only be another operation while child elements can be either an operation or a process.
Process	Focus on accomplishment of work tasks and flow of resources.	It can only have one parent and no children. The process element defines the lowest level of the WBS.

In the second step a resource library for the project is created. All resources required on the project are initialized in this library. Details like resource name, quantity allocated to the project, variable cost and fixed cost of the resource are provided. The resources can be defined at the project level or at any level of the WBS, which makes resource definition an integral part of the development of the project WBS.

In the third step the identified operations are sequenced. The modeler schedules various operations based on the construction logic and sequence. For this purpose HSM provides serial, parallel and cyclic links. An operation can also be designated as a hammock operation by simply specifying a start time. The definition of links is synonymous with PDM network logic constraints except that they could be at any level of the WBS and advanced relations such as cycling of work components can be used. This provides versatility to HSM and makes it equally applicable on building projects or linear projects like pipeline construction. Figure 1-2 illustrates the operation sequencing of the hypothetical building project. Table 1-2 defines these links and their interpretation.

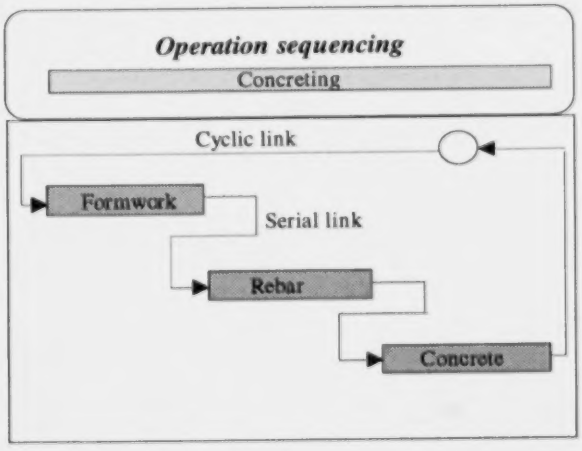


Figure 1-2: Illustration of operation sequencing

Table 1-2: Operation sequencing links

Link	*Interpretation*	*Usage*
Serial	Construction logic dictates that operations can be implemented in series one following the previous.	A predecessor operation is linked to a successor using a serial link. A lead or lag (in days) can be assigned for any serial link.
Parallel	Construction logic allows the operations to be performed simultaneously.	Two operations that can be implemented in parallel are linked using a parallel link. One operation is termed as the predecessor and the other is the successor. A lag in days can be attached to the parallel link.
Cyclic	Construction logic requires that one operation or a group of operations are performed a given number of times.	One operation or a group of serially linked operations can be repeated by linking the last operation to the first operation using a cyclic link. A counter is attached to the cyclic link to determine the number of repetitions.
Hammock	Construction logic is such that the operation is not constrained by any other operation in the project.	A single operation is isolated from other operations by defining a hammock link. The hammock link is defined in terms of days relative to the start time. Once an operation is defined as hammock no other links can be assigned.

The fourth step involves development of process models for the defined processes in the WBS. The process models are developed using the CYCLONE modeling methodology (Halpin, 1977) with enhancements and additional modeling elements to allow project simulation. The enhancements to the CYCLONE method include addition of resource manipulation elements, process inter-linkage and inter-process constraining relations. The additional nodes developed as part of the HSM modeling framework are shown in Table 1-3.

Table 1-3: Process modeling elements in HSM

Modeling element	*Description*
Allocate Resource Node	Resources defined in the resource pool at the project level are attached to the process models using the Allocate Resource Nodes. A single resource or a resource combination can be allocated at this node. The node captures the required quantity of resources from the resource pool before a work task can be started.
Free Resource Node	This node works in conjunction with the Allocate Resource Node. A single resource or a resource combination can be released back to the resource pool using this node. It is essential to have a corresponding Allocate Resource Node for every Free Resource Node. In the absence of the Free Resource Node the resources captured at an Allocate Resource Node are not released back to the resource pool.
Predecessor node	This node in conjunction with the successor node is used to define process interdependencies. A predecessor node releases an entity to a succeeding process on the completion of a defined number of cycles of the current process.
Successor node	The successor node works in conjunction with the predecessor to define process inter-dependencies. It receives the entity released by the predecessor node in the preceding process. After receiving the entity the successor node allows the start of the succeeding process.

In HSM the modeler can allocate resources either at the operation or process levels. For modeling a resource that is assigned to an operation without a detailed identification of the work tasks for which the resource is required, the modeler allocates the resource to that operation. In this case the resource is automatically allocated to all descendent operations and processes below this operation in the hierarchy. The modeler can also attach the resources defined in the resource pool to the process models. In both cases it is feasible to allocate the same resource to more than one operation or process. Figure 1-3

illustrates process models for two processes that share common resources and are linked to each other.

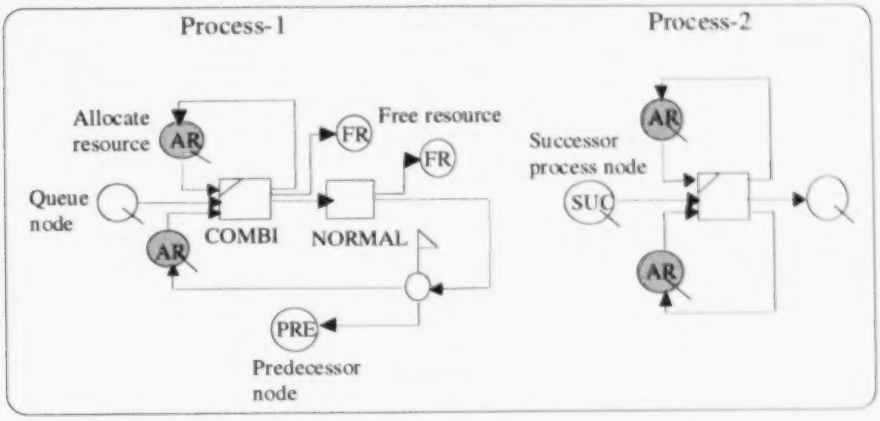


Figure 1-3: Illustration of process models in HSM

The final step in the HSM analysis is to extend all models to a common level where simulation will take place. Computer implementation of the method automatically accomplishes this as demonstrated in Figure 1-4. Currently the models are translated into a general purpose simulation language to facilitate experimentation.

1.6 PROTOTYPE SYSTEM FOR HSM

An essential ingredient of HSM is its rich graphical user interface that allows a simple method for development of the project plan. A prototype has been developed by the author which illustrates the potential benefits of the proposed method and also allows testing of the modeling concepts. Internally, HSM is implemented in four modules which include a graphical model building module, translation module, simulation module and report generation module.

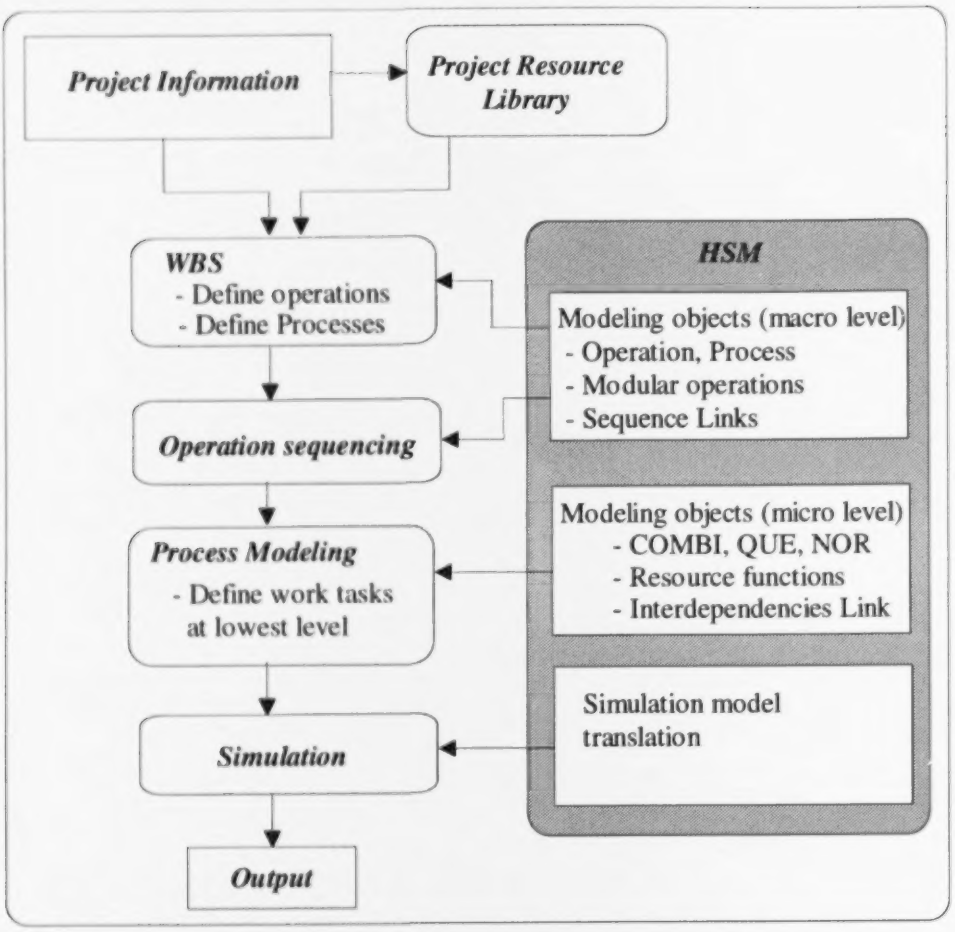


Figure 1-4: Internal working of HSM

The graphical model building module has been implemented in an object-oriented environment. The graphical elements (operation object, process object and process modeling objects) were developed as "control classes" in Visual C++ (Microsoft, 1993b). These "control classes" are then loaded in Visual Basic (Microsoft, 1993a). The graphical model building module is developed in Visual Basic using these objects. The translation module for the prototype system was developed using Visual Basic and is internally linked to the graphical model building module. The function of the translation is to automatically

combine the information provided in the WBS, operation sequencing and process models to formulate one discrete event model. Currently the system translates and forms the basic model in SLAMSYSTEM simulation language. In the prototype system SLAMSYSTEM is being utilized externally as the simulation module.

1.7 THESIS ORGANIZATION

The following chapters describe the various parts of the HSM framework developed in this research. Chapter 2 describes the modeling concepts of the method. It provides the modeling framework for developing a project WBS, defining resources, linking operations and developing process models. Chapter 3 describes the implementation details and internal structure of the computerized tool developed to support the proposed method. Chapter 4 illustrates the concepts and benefits of the proposed method by using an actual project example. Chapter 5 summarizes the simulation based method, addresses the significance of the work, and outlines future research directions in the area of simulation-based planning of construction projects.

1.8 CONCLUDING REMARKS

This chapter provided a brief introduction to the area of planning of construction projects. A statement of the research problem and research objectives was provided after the initial introduction. A brief overview of different project simulation modeling methods was described in order to illustrate the state of the art in this area. An overview of the Hierarchical Simulation Modeling method developed by the author illustrated the modeling framework required for project level simulation.

Chapter 2: HSM Modeling Concepts

2.1 INTRODUCTION

Planning of construction projects using the Hierarchical Simulation Modeling (HSM) method requires the accomplishment of the following steps:

1. Developing the project breakdown by identifying the operations and processes.
2. Defining the resources allocated to the project.
3. Defining the logical relations between the various project components.
4. Developing the process models using the CYCLONE simulation method.
5. Analyzing the simulation models and generation of the results.

The HSM framework developed by the author provides a guideline for performing these tasks. In this chapter a description of the modeling concepts of HSM, that guide the plan development, is provided. The specifications to be followed for the preparation of the project plan are discussed under three different sections. The first section covers project scoping using the hierarchical and modular breakdown structure and resource definition. In the second section the rules for sequencing the identified operations are described. Specifications for developing the process models are described in the third section.

HSM uses a symbolic graphical format for the development of a project plan. Modeling elements for the development of WBS, process models and links for operation

sequencing are available to the modeler.

2.2 SPECIFICATIONS FOR PROJECT SCOPING

The scoping of a project in HSM uses the following concepts:

- Hierarchical work breakdown structure.
- Definition of modular project components, and
- Resource definition.

The objective behind project scoping is to divide and streamline the project information into manageable units. This essentially involves identification of the work to be performed on the project, identification of repetitive work components (if any) and resource requirements for the project. Description of these concepts is provided in the next three sub-sections.

2.2.1 Work breakdown structure

HSM utilizes the concept of "hierarchy" of work involved on a construction project that was formulated by Halpin and Woodhead (1976). Under this scheme, the work on a project is divided into the following categories:

1. Project level is the highest level and the focus at this level is on "gross project attributes" like project cost, schedule, resources, material requirement and other management issues.
2. Operation level is the next level in the hierarchy where the focus is on construction methods and implementation strategy.
3. Process level is the third level with focus on the basic technological sequence of work.

4. Activities or work tasks form the fourth level of the hierarchy with focus on the physical work required for the project.

Figure 2-1 provides a graphical illustration of the above described hierarchy.

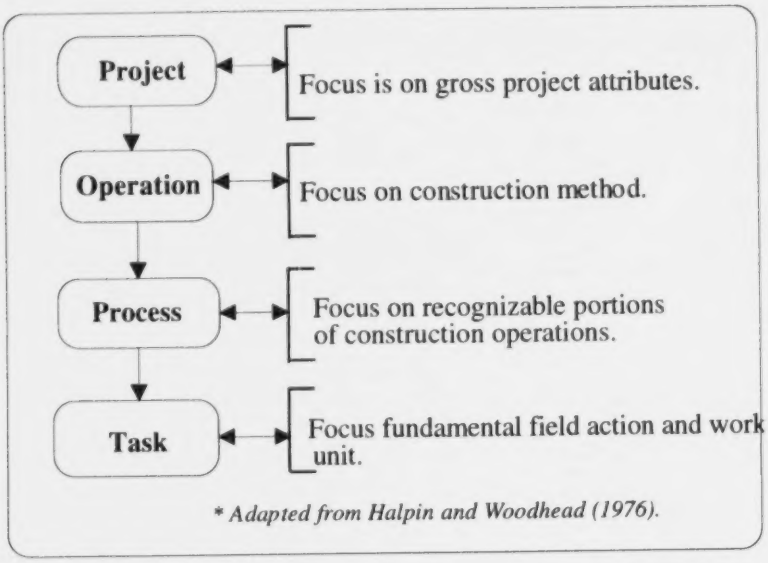


Figure 2-1: Hierarchy of work on a construction project

The above breakdown of a project is commonly referred to as the work breakdown structure of the project (WBS). WBS has been studied by various researchers and practitioners in the past. For a detailed discussion of this topic refer to research by Elmore 1976, Murphy 1981, Halpin et. al. 1987 and Halpin and Riggs 1992.

The first step in the HSM modeling process involves development of the project work breakdown structure. HSM uses three constructs namely: "project," "operation" and "process" elements for performing this task. Each project contains a root element called project-element. All other elements originate from this element and are its descendants. Operations in the WBS descend from the project or other operations (terms used to describe the relationships in the WBS include parent, child and descendent and have their

usual familial meanings). An operation element can have more than one child element attached to it. Children of an operation element can either be other operations or process elements provided that they are of the same category (i.e. all operations or all processes). Process elements form the lowest level of project breakdown. These elements cannot have any descendants. They have one operation element as their parent. A project must have process elements in its work breakdown.

A large project or operation could be divided into various levels of detail in a top-down fashion. Figure 2-2 shows the WBS of a sample bridge project. In the illustration the bridge project is divided into five different levels. Level 0 corresponds to the project level. The element named "bridge project" is the "project-element" or the "root-element" for this WBS. Level 4 is the lowest level of the project WBS. This level models the various processes that are involved in the project. In Figure 2-2 "formwork", "rebar" and "concreting" are process-elements. Level 1, 2 and 3 in Figure 2-2, model various operations that are involved in the project. Each of these levels is composed of "operation-elements".

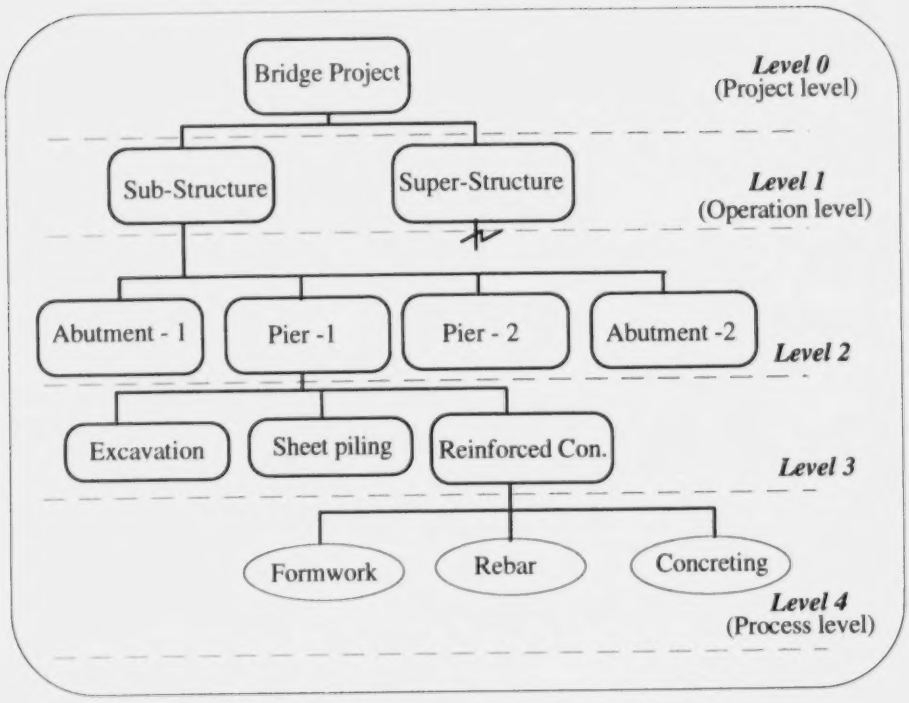


Figure 2-2: Work breakdown structure of a bridge project

A project can be broken down into different levels depending on its complexity. In the illustration, the bridge project has been broken down into five different levels. A project must have at least two levels in its breakdown, one being the project itself and the other being the process level. If the project being modeled is not complex then it need only have level 0 and level 1 where level 0 will be project level and level 1 will be process level as shown in Figure 2-3(a). On the other hand, for a complex project it may be necessary to have intermediate levels that break the project into operations. Hence, a work breakdown structure may consist of a project level, one or more intermediate operation levels and one process level as shown in Figure 2-3(b).

The best approach on a construction project is to divide the project into operations and processes based on the physical sub-divisions of the project (e.g., floors of a building,

spans of a bridge, etc.). Within each operation further sub-division can be achieved by physical or responsibility groups.

A project may be divided into operations such that different operations have different levels of detail. In Figure 2-3(b) operation "Opr A" can be divided into level 2 while the operation "Opr C" to n+1 levels. Therefore, HSM allows division of operations into different levels of detail.

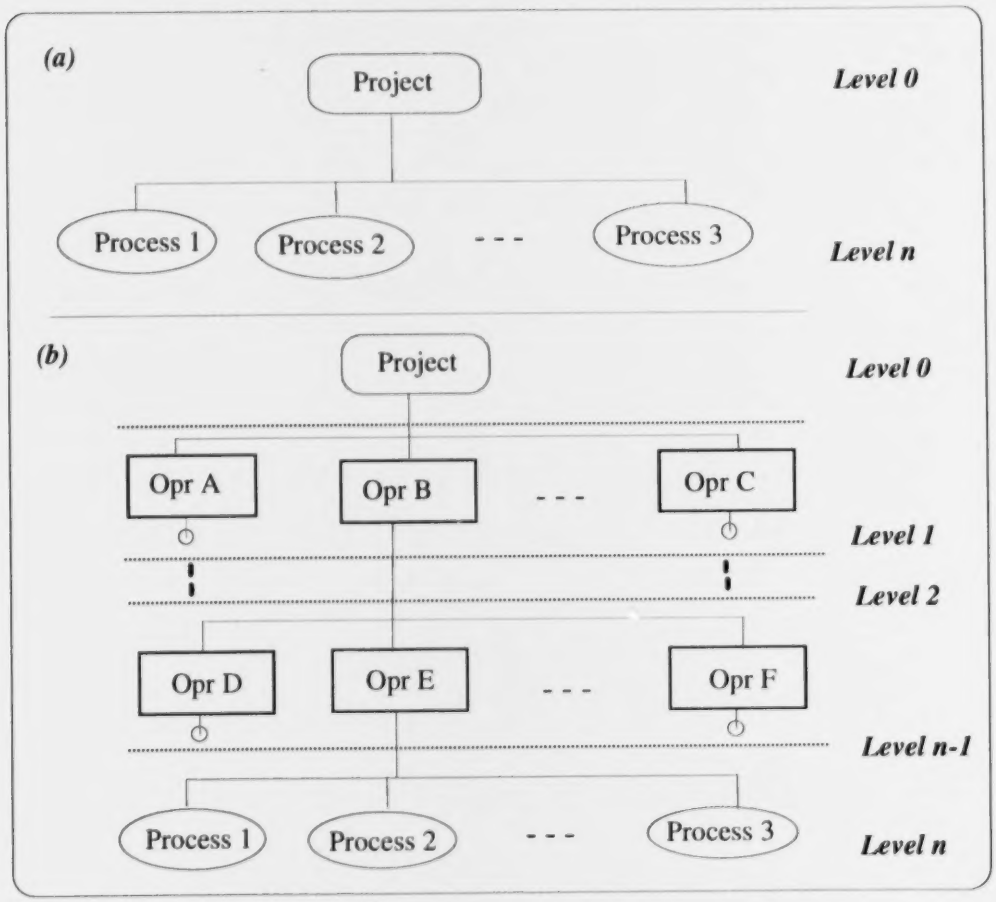


Figure 2-3: WBS with different levels of detail

2.2.2 Modularity concepts

HSM enhances the modeling process of a construction project using modular concepts.

Many repetitive technological components can be found in construction projects. These recurring units of work (operations as well as processes) can be combined in modules and referenced in the WBS as needed. This is referred to as modular components in HSM. The concept of modularity for simulation models was introduced by Zeigler (1987) and Luna(1990).

Modularity is especially useful for projects that involve repeated modules of work. Figure 2-4 illustrates modularity concepts for planning a high-rise building.

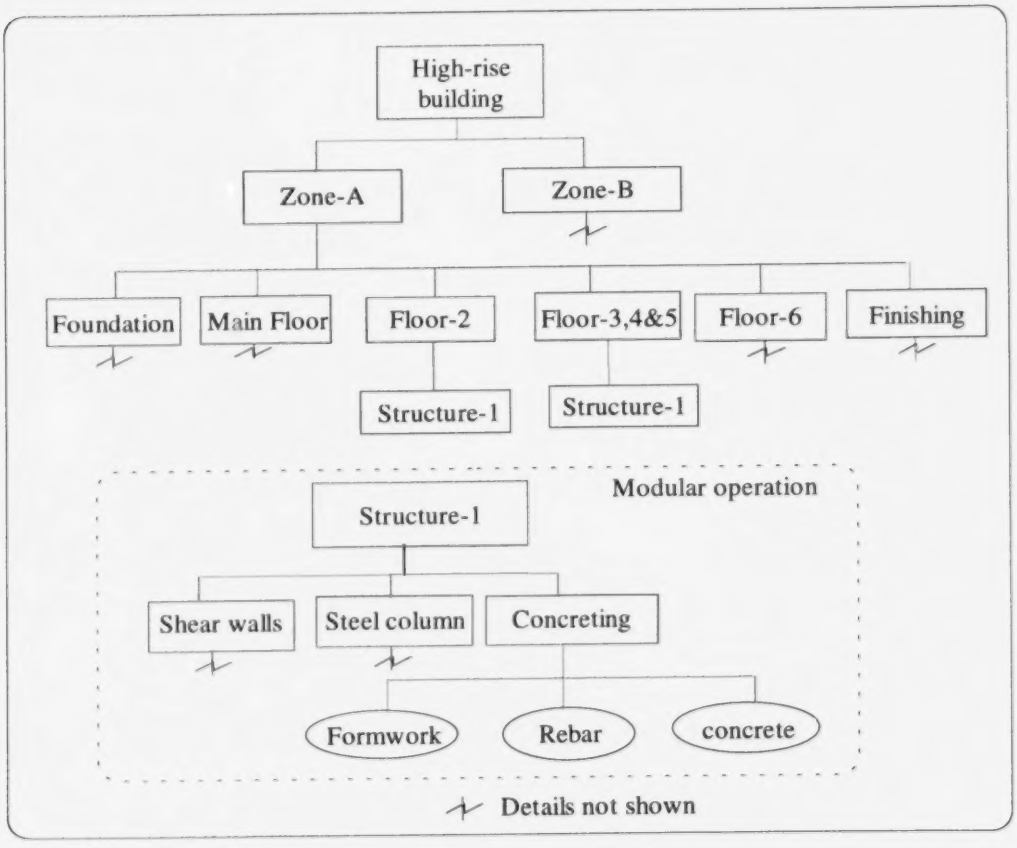


Figure 2-4: Modularity in Project WBS

Figure 2-4 shows the WBS in which a modular operation is defined and referenced by

the "floor operation". The modular operation ("Structure-1") has its own WBS definition. Since the construction method for the Floor 2, 3, 4 and 5 is assumed to be the same, they all reference this modular operation, thus greatly simplifying the WBS.

Figure 2-5 shows a partial WBS for another hypothetical project. This figure illustrates the use of modular processes.

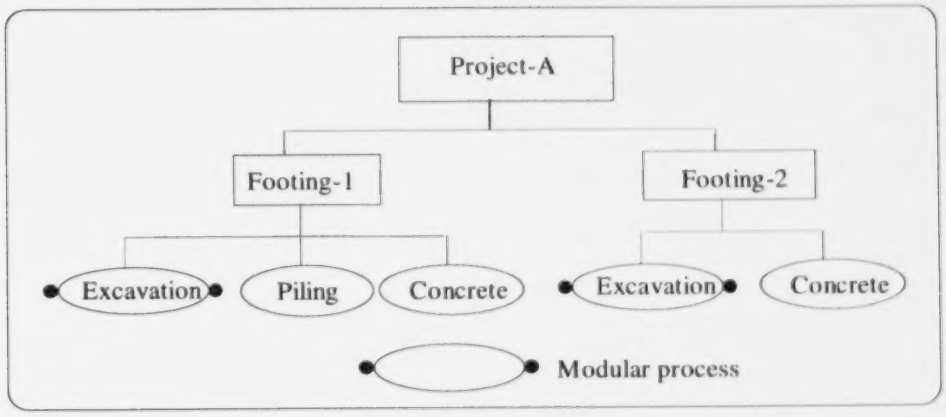


Figure 2-5: Modular processes in project WBS

The project involves construction of two types of footings, footing-1 and footing-2. In footing-1 after excavation, a piling process is required before the concrete can be placed, whereas the footing-2 requires only excavation and concrete processes. It is assumed that the construction method for both excavation processes is the same, a modular process was defined and was referenced by both footing operations in the project WBS. The modular process is depicted in the figure by an ellipse with two dots. The benefit of the definition of the modular process apart from reducing the complexity of the WBS is that the modeler is required to define the process model (further discussed in the section 2.4) only once. This is especially useful because processes often remain unchanged depending on the technology.

2.2.3 Resource Definition

HSM provides detailed resource modeling capabilities for a given project. Following the natural approach to resource assignment on a project, HSM defines resources at the project level. These resources are then allocated to various tasks as needed during the simulation experiment. Tasks compete for resources which are dynamically updated throughout the analysis.

Resource attributes are defined for each individual resource to facilitate its logical allocation to the required tasks, resolve conflicts through priorities and to determine costs. As with modularity, for a given project type (e.g. heavy or highway construction) resources can be defined once by the contractor and may be used for planning new projects. Resource libraries such as the one shown in Figure 2-6 are portable from project to project. They may be defined in a master library on an as needed basis.

Resource library

ID	*Resource Name*	*Quantity*	*Fixed Cost ($/hour)*	*Var Cost ($/hour)*
1	Crane	1	102.00	59.00
2	Backhoe	2	50.00	25.00
---	---	---	---	---
n	Diesel Hammer	1	100.00	65.00

Figure 2-6: Illustration of resource library

2.3 OPERATION SEQUENCING

Operation sequencing allows the modeler to specify the implementation strategy for the project being planned. This information is important to determine the implementation sequence of the underlying processes. On the basis of the construction logic, technology,

strategy and resource availability various operations can be sequenced in different ways.

To illustrate the concept of operation sequencing a simplified WBS for a project is shown in Figure 2-7. There are three operations defined at level-1 and four operations at level-2. Operations Opr-12 and Opr-11 are children of Opr-1, while operation Opr-22 and Opr-23 are children of Opr-2. The remaining details are not shown in the figure for brevity. The sequencing of these operations is illustrated in Figure 2-8. Figure 2-8(a) provides sequencing for operations at level-1 and Figure 2-8(b) provides sequencing for operations at level-2. Since operations at level-1 have a common parent, a combined sequencing is provided. On the other hand for level-2 there are two different sequencing diagrams, one for the operations with Opr-1 as their parent and one for operations with Opr-2 as their parent. From this example it can be seen that sequencing of operations has to be provided for each level of the WBS and operations at a particular level have to be grouped together based on their parent operation. The first step in sequencing a group of operations is to provide one or more start operations. The remaining operations in the group are then linked to the starting operations to provide a complete sequencing.

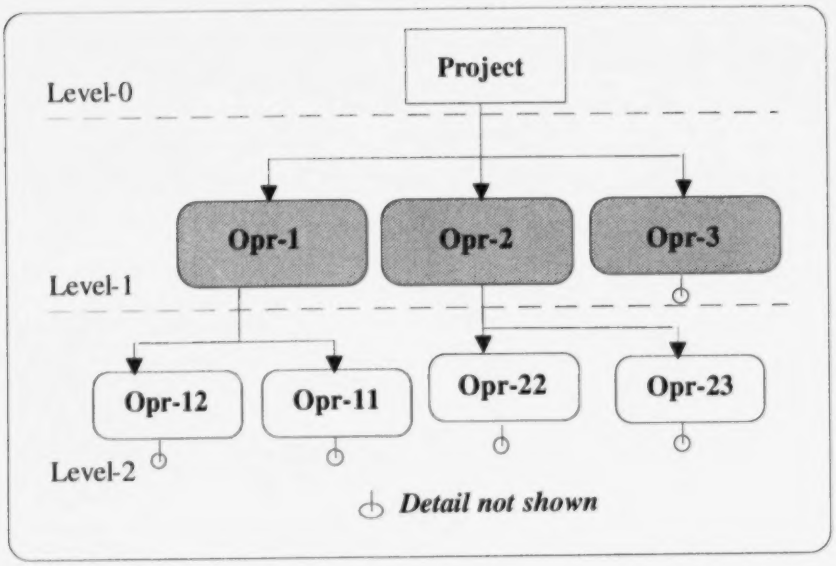


Figure 2-7: Operations with common parent

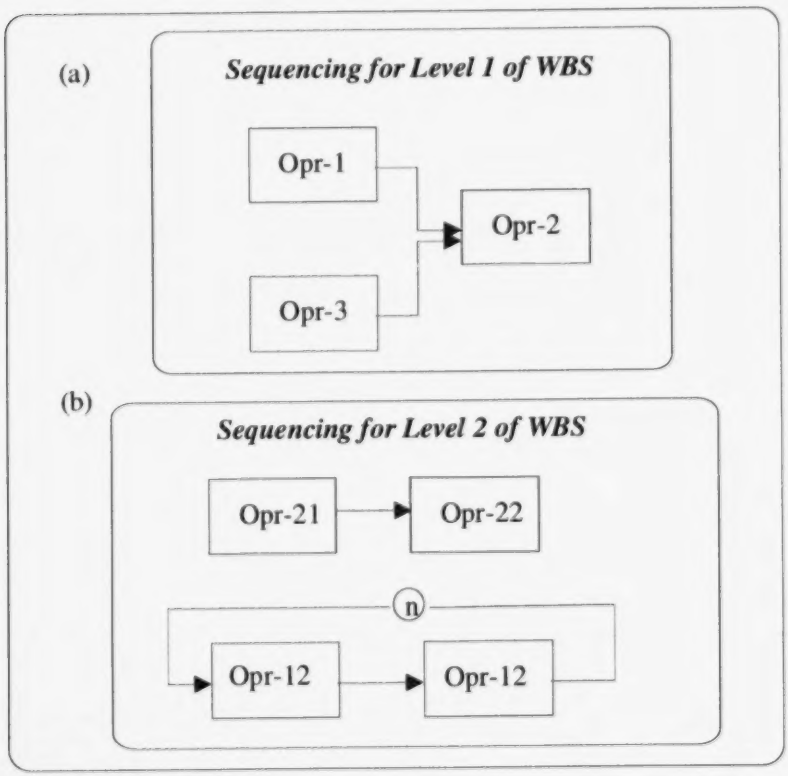


Figure 2-8: Operation sequencing diagrams

2.3.1 Types of links in HSM

HSM provides the following four types of relationships:

Serial relationship: This relationship models situations where an operation can only start after the completion of another operation. A serial relationship between operations is equivalent to a finish to start relationship between activities in PDM. This relationship can have a value of finish to start lead or lag.

Parallel relationship: A parallel relationship between operations basically denotes that the two operations may proceed simultaneously. This is equivalent to a start to start relationship in PDM.

Cyclic relationship: Cycling of operations is a key feature of HSM that makes it versatile and allows efficient planning of linear and repetitive construction projects. This enhancement provides a tool that is not available in network based planning methods. The cyclic relationship is explained with the help of a sample linear construction project. Figure 2-9 shows the WBS and operation sequencing for a pipeline project which is 100 km in length. It is assumed that the pipeline project can be broken down into three operations. The traditional scheduling methods could require the identification of a unit of construction, which in this case would be 1 km, and then scheduling the three basic operations in 100 such units. In the proposed planning method this can be simplified by using the cyclic relationship between operations as illustrated in Figure 2-9.

There are two possible scenarios which can be applied to sequence operations. In the first scenario (A) the three operations are first linked serially using the serial relationship and then a cyclic relationship is used between the "backfilling operation" and the "earthwork operation". In the cyclic relationship a "cyclic counter" is attached. This node

determines the number of times the operation or a group of operations have to be repeated.

In the second scenario (B) the three operations are first serially linked and the cyclic relationship is used to perform the repetition of the individual operations based on the information provided in the cyclic counter. In option B the excavation operation is started in the first section of the pipeline. Once the excavation operation is completed the piping operation begins in the first section while the excavation process repeats itself in the second section. The same logic is followed for the backfilling operation for the entire length of the project.

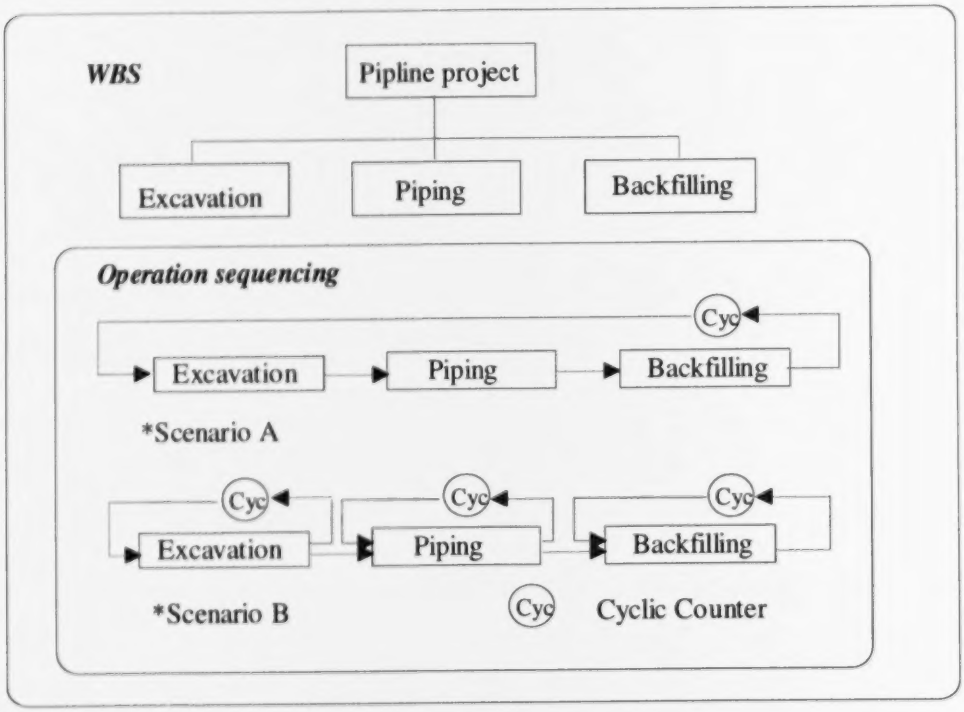


Figure 2-9: Cyclic relationship

Hammock operations: Some operations are independent of all other operations and as such they neither have a serial nor a parallel relationship with other operations. Any operation that does not have a serial or parallel operation is termed as a hammock operation. A hammock operation is defined by its start point only. On a construction project a hammock operation can be considered as a sub-project with no constraints from the remaining project operations.

2.3.2 Rules governing the use of HSM relationships

The four links described here provide the modeler with the flexibility to model different scenarios normally experienced on construction projects. However, in HSM certain rules govern the use of these links to avoid conflicts and reduce redundancies. The governing rules are as follows:

1. Operations with common parents are sequenced as one group. For example, operations at level-2 in Figure 2-7 are divided into three groups for operation sequencing. The modeler is not allowed to link Opr-12 with Opr-22 or Opr-23 because they belong to different parent groups.
2. Once an operation has been defined as hammock, no other links can be provided for that operation.
3. For connecting more than one operations by a cyclic link the modeler has to first connect them serially. For example in Figure 2-9 for scenario "A", serial link between the "excavation", "piping" and "backfilling" operations is defined along with the cyclic link.

2.4 PROCESS MODELS

Process models are used to model the tasks in a project at the lowest level. The process models are developed using CYCLONE modeling elements. Three enhancements have been made to the CYCLONE method to allow project level simulation. These enhancements are as follows:

1. modeling of resources which are defined at the project level.
2. modeling of process interdependencies.
3. resource identification.

The modeling philosophy adopted is the CYCLONE methodology. For a detailed discussion of CYCLONE and its applications, refer to Halpin (1973), Halpin and Woodhead (1976), and Halpin and Riggs (1992).

The CYCLONE modeling elements are shown in Figure 2-10 with a brief description in Table 2-1.

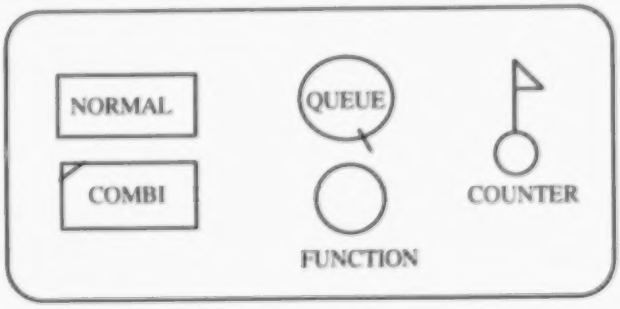


Figure 2-10: CYCLONE modeling elements (source Halpin, 1990)

Table 2-1: Rules for developing CYCLONE models (source Halpin, 1990)

Element	Rule
NORMAL	This node represents a non-constraint work task similar to a work bay with an infinite number of servers.
COMBI	This node represents a work task constrained by the availability of more than one type of resource. A COMBI is processed when all required resources are available.
QUEue	A node where idle resources wait. A resource arriving at a QUEue node will stay in the node until a COMBI is ready to process it.
FUNction	New elements can be created at this node.
COUnter	A node that keeps track of the number of times a unit passes it. It does not alter any of the resources or their properties.

2.4.1 Modeling resources at the process level

The resources that are defined at the project level are used to perform the tasks that are identified in the process models. Resources can either be linked to one process or to a number of processes. When the resource is assigned to more than one process a priority is provided to each process which determines the allocation of the resource during simulation.

Traditionally, modeling of resources in CYCLONE is done by initializing resources at appropriate QUE nodes in the process model. Though this strategy provides a simple method of modeling resources for the individual process, it does not facilitate modeling of shared resources across processes. In order to overcome this modeling difficulty two modeling elements are defined. The first element is called an Allocate Resource Node and is an extension of the QUE node. In a process model, a COMBI that requires a resource

defined in the resource library is preceded by the Allocate Resource Node. This node acts as a special QUE, that checks the availability of the specified resource. The Free Resource Node belongs to the FUNCTION base class. The Allocate Resource Node and Free Resource Node normally work in conjunction. A Free Resource Node is used for every Allocate Resource Node if the resource has to be released back to the resource library. However, it is feasible to provide an Allocate Resource Node without a corresponding Free Resource Node. In this case the resource will be allocated and will not be released back to the resource library. As the name suggests the Allocate Resource Node captures the required resource from the project level resource library and allocates it to the process for completion of the work tasks. Similarly the Free Resource Node frees the resource captured at the Allocate Resource Node back to the common library. The graphical representation and internal structure of both nodes is provided in Figure 2-11.

Allocate Resource Node

Free Resources Node

Node ID	Process ID	Follower	Resource ID	Quantity	Priority

Figure 2-11: HSM resource nodes

Figure 2-12 shows an illustration of the use of the Allocate Resource Node and of the Free Resource Node for a process model. In this Figure it can be seen that before the entity arrives at the COMBI node, the Allocate Resource Node tries to allocate the designated resource. If the resource is available then the entity is processed otherwise the entity waits in the Allocate Resource Node in a fashion similar to a QUE node. It should be noted that the Allocate Resource Node (which is a special type of QUE) replaces the

QUE node before the COMBI in this particular scenario.

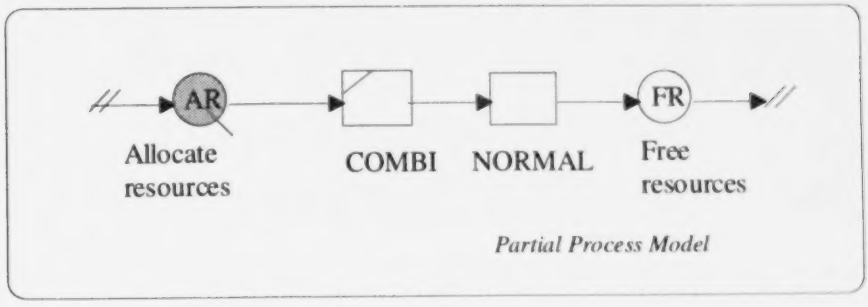


Figure 2-12: Illustration of resource nodes in a process model

The resource nodes can be used to allocate and free multiple resources. For example if a work task requires 1 crane, 2 trucks and 3 operators, then the same Allocate Resource Node can be used to capture these resources from the resource library. Once an Allocate Resource Node is used to capture multiple resources a corresponding Free Resource Node can be used for freeing all the resources that were captured. This feature greatly reduces the complexity of the model.

Another important enhancement for modeling of resources is that the Allocate Resource Node can be used to assign alternate resources for the process. This scenario is common on a construction site. For example a hauling operation can be performed by trucks of 10 m^3 capacity (type A) or trucks of 5 m^3 capacity (type B) depending on the availability. This scenario is modeled by first defining truck A and truck B as separate resources in the resource library and then using the Allocate Resource Node to select the truck that is available at the time of scheduling the work task in the process. The Free Resource Node will automatically free the appropriate resource.

2.4.2 Modeling process interdependencies

Two or more processes that have a common parent operation in a project can be linked

together using the process interdependency links. In traditional planning methods, since the work tasks are defined at a common level these relationships are implicitly modeled by activity relationships. In HSM the modeler defines separate process models and links them explicitly. The process interdependency links work in conjunction with the operation sequencing defined at the operation level and ultimately provide the linkage between all the processes.

HSM utilizes the concept of class IV control structure as defined in Halpin and Woodhead, 1976 "*the control of one section of the system depends on the completion of work tasks in another section, it is essential in such instances that a message indicating that certain events have occurred must be sent*". Figure 2-13(a) shows a general form of class IV control structure as depicted by Halpin and Woodhead, 1977 and its adaptation in the proposed method.

In Figure 2-13(a) it is shown that process-2 can only start when a release entity is available at the control QUE that links it with process-1. In the proposed planning method a similar approach is required to link the processes. The link is achieved using two elements as shown in Figure 2-13(b).

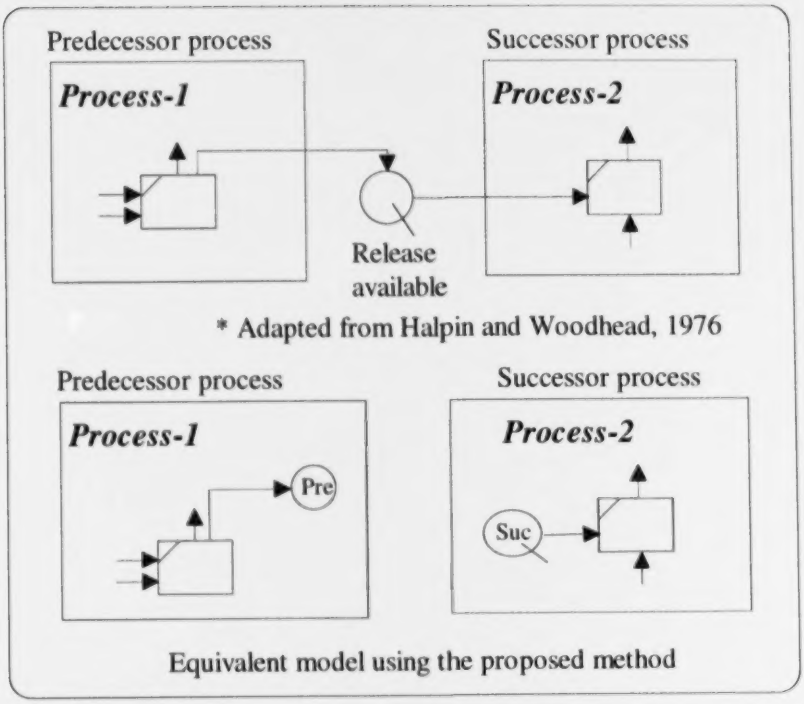
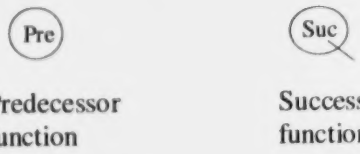


Figure 2-13: General form of Class IV control structure

Two elements explicitly model the inter-process dependencies namely: predecessor and successor elements. Figure 2-14 shows a graphical representation of both elements.



Predecessor ID	Successor ID	Quantity/Cycle	Number of entities per release

Figure 2-14: HSM process interdependency elements

These elements are internally represented by defining the predecessor process, successor process, quantity and number per release. This representation is a table in a relational database as illustrated in Figure 2-14. The predecessor element belongs to the

FUNCTION base class and is used in the predecessor process. It can be placed after a COMBI, NORMAL, CONSOLIDATE, FREE RESOURCE or a FUNCTION COUNTER (precedence rules for the HSM modeling elements are described in section 2.4.3). Multiple processes can be linked from the same predecessor function. The predecessor function can release entities to one or more successor processes.

The successor element belongs to the QUE node class and replaces a simple QUE that initializes control units, consumable materials or construction components in the successor process. A process can have one or more successor elements, but these could be linked to different processes. Control to the successor element is only released when the required quantity or cycles of the predecessor process have been completed. On the basis of this feature a modeler can specify two types of relationships between processes namely:

1. A successor process starts after completion of predecessor process.
2. A successor process starts after partial completion of predecessor process.

Consider the Pier-1 operation of the bridge example explained earlier. It has three underlying processes namely: "earthwork", "piling" and "pier-shaft". From the operation sequencing defined by the modeler, HSM automatically determines that the earthwork process starts first, followed by the piling process, and then the pier-shaft process. This scenario is illustrated in Figure 2-15.

The excavation process requires one backhoe and five trucks that have been modeled by two sets of the Allocate Resource Node and Free Resource Node. The process requires excavation of 100 units of earth that was modeled as a QUE with initialization.

The function counter in the earthwork process model is followed by the predecessor function. The specification provided by the modeler is such that 6 entities (piles) are released after 100 units of soil have been excavated and moved. These entities released in the successor element are used in the piling process. For illustration a dotted line has been shown between the predecessor and successor element.

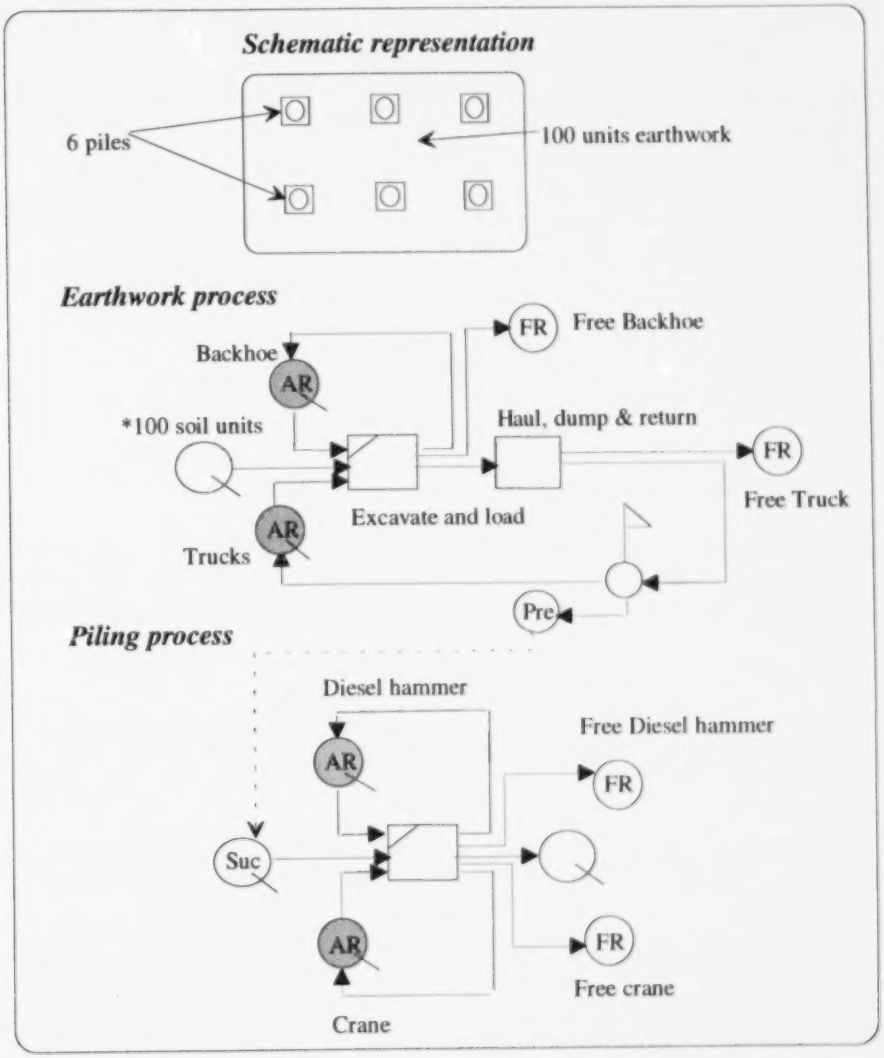


Figure 2-15: Illustration of process inter-dependency

2.4.3 Precedence rules for process modeling elements

The HSM process modeling elements including the CYCLONE modeling elements are allowed to be used in a flexible manner for the development of process models. The modeler develops process models by assembling these modeling elements in different arrangements. In the development of process models, however, certain precedence rules have to be followed. These rules are summarized in Table 2-2. The letter in each field of the row in the table indicates feasibility of an element "A" (indicated to the left of each row) preceding an element "B" (indicated at the top of each column). The letter "M" in a row indicates that it is mandatory to provide the element "B" as a successor to the element "A". "I" indicates that the predecessor and successor relationship between the elements "A" and "B" is allowed and "N" means that the element "B" is not allowed to follow the element "A". For example, a COMBI element cannot be followed by another COMBI element. This is denoted in the Table 2-2 by the letter N in the first row and first column. Similarly, it is mandatory to have COMBI as a follower of a QUE element. From the table it can be deduced that a "Successor" element cannot have any predecessors and a "Predecessor" element cannot have any successor.

Table 2-2: Precedence table for HSM process modeling elements

A \ B	Work tasks		QUE Elements			FUNCTION Elements			
				AR	Suc			FR	Pre
	N	I	I	I	N	I	I	I	I
	N	I	I	I	N	I	I	I	I
	M	N	N	N	N	N	N	N	N
AR	M	N	N	N	N	N	N	N	N
Suc	M	N	N	N	N	N	N	N	N
	N	I	I	I	N	N	I	I	I
	N	I	I	I	N	I	N	I	I
FR	N	I	I	I	N	I	I	I	I
Pre	N	N	N	N	N	N	N	N	N

M = Mandatory or required, I = Immaterial, N = Not allowed

Using the above described constraints on the arrangement of elements the modeler can develop the process models for the processes identified in the project WBS.

2.5 CONCLUDING REMARKS

This chapter provided the description of the modeling concepts of HSM. The rules and specifications required while developing the project WBS, defining resources, sequencing operations and developing process models were described. Further explanation of these modeling concepts is provided in Chapter 4 with the help of an example project. The implementation details and use of the prototype system is provided in Chapter 3.

Chapter 3: Computer Implementation of HSM

3.1 INTRODUCTION

The Hierarchical Simulation Modeling (HSM) method provides the modeling framework for simulation based planning of construction projects. A prototype computerized modeling environment was developed to allow modeling under the HSM framework. Various issues were considered before the implementation of HSM. Those included the following (AbouRizk and Sawhney 1994):

- The tool should be flexible to model project information for different types of construction and should allow quick and efficient project scoping.
- It should model all the aspects of the project including shared resources and process inter-dependencies and should provide support for modularity and reusability of its components.
- It should perform simulation in the background so as not to intimidate the modeler.

This chapter addresses the implementation of HSM.

3.2 BACKGROUND

In recent years various researchers and simulation practitioners have highlighted the use of object oriented concepts in simulation. Roberts and Heim (1988) state that "*there is little doubt that future simulation languages will incorporate more ideas from object oriented perspective, especially as a means of extending the language to a wider variety of applications*". Rothenberg (1986) states that "*object-oriented simulation provides a*

rich and lucid paradigm for building computerized models of real world phenomena". In recent literature, researchers have focused on various features of object oriented programming that can be beneficially adopted in simulation. Liu and Ioannou (1992) demonstrated the graphical modeling strategy that can be achieved using object oriented implementation through the development of COOPS. Using an object oriented design they created simulation modeling elements similar to CYCLONE modeling elements. More recently, Oloufa (1993) performed initial experiments on the use of object-oriented simulation for construction operations.

The hierarchical and modular modeling approach that is effective in simulating complex systems can be closely associated with object-oriented programming. The works of Zeigler (1984) and Luna (1992) provide some insight in the area of hierarchy and modularity of simulation models.

Event driven programming is a relatively new technique of programming in which a program is developed by responding to actions taken by the user or system driven events (Microsoft, 1993a). In traditional or "procedural" programming the application itself controls the execution of different portions of the code. In event driven programs, a user's action or system event executes an event procedure. This is the essence of graphical user interfaces and event-driven programming. By using object oriented concepts and event driven programming in conjunction a graphical user interface can be created in which the objects respond to events created by the user.

3.3 OVERVIEW OF HSM

The internal design of HSM is such that there are four distinct modules developed to accomplish different tasks as illustrated in Figure 3-1. The four modules are "model definition", "translation", "simulation" and "report generation". The model definition module provides a graphical environment for the development of a project WBS, definition of resources, sequencing of operations and the development of process models. The function of the translation module is to compile and translate the plan information provided by the user and convert it into a combined simulation model. This model is then processed by the simulation module and reports are generated through the report generation module.

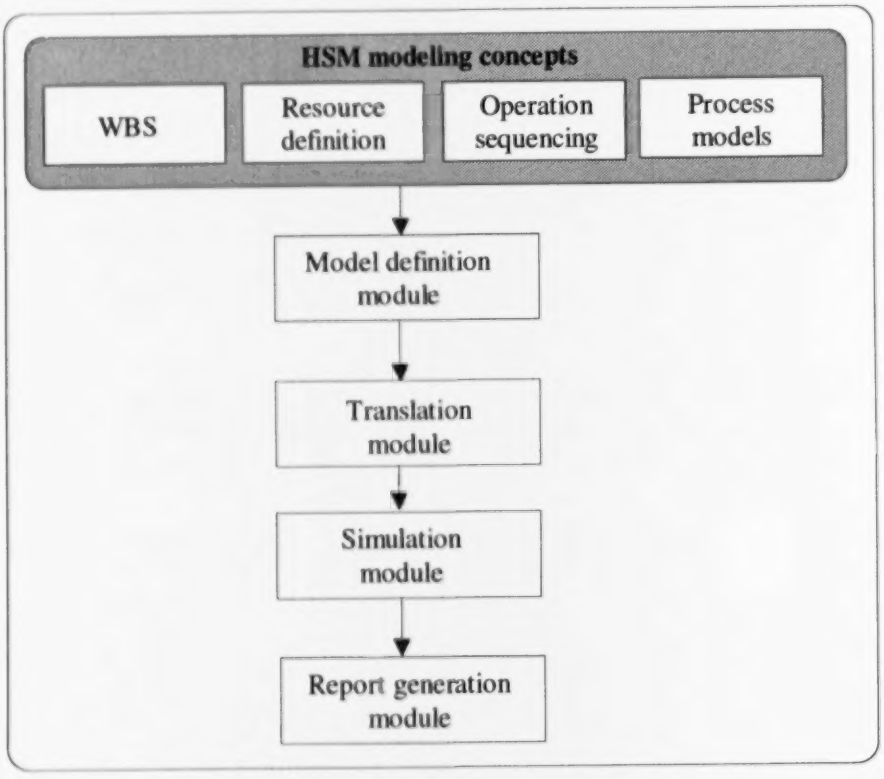


Figure 3-1: Internal structure of HSM prototype

3.4 IMPLEMENTATION OF HSM

An interactive graphical environment that solicits the project information in a natural and logical manner was deemed to be a major factor in the success of HSM as a planning tool. This goal was achieved through the use of object oriented concepts and event driven programming in the implementation process. A project plan is developed by instantiating (creation of an instance of a pre-defined object) the modeling elements provided in HSM. Internally these modeling elements have been implemented as "objects". These objects have two groups of properties namely: "graphic" and "simulation" properties. The graphic properties of these objects assist in the working of the graphical user interface while the simulation properties assist in the development of simulation models. The objects were first developed in an object oriented programming environment and were then imported to an event driven programming environment where the graphical user interface was programmed. With this strategy, properties of the modeling objects were programmed to produce the desired behavior. Figure 3-2 illustrates this concept graphically. For the development of the HSM prototype, Visual C++ (Microsoft, 1993b) was used as the object oriented programming environment and Visual Basic (Microsoft, 1993a) was used as the event driven programming environment. Visual Basic provides a more intuitive development environment that allows "non-programmers" to contribute to the system with less training than required with C++. This strategy was found to be suitable for the prototype where the construction modeling aspects were more important than the program efficiency itself.

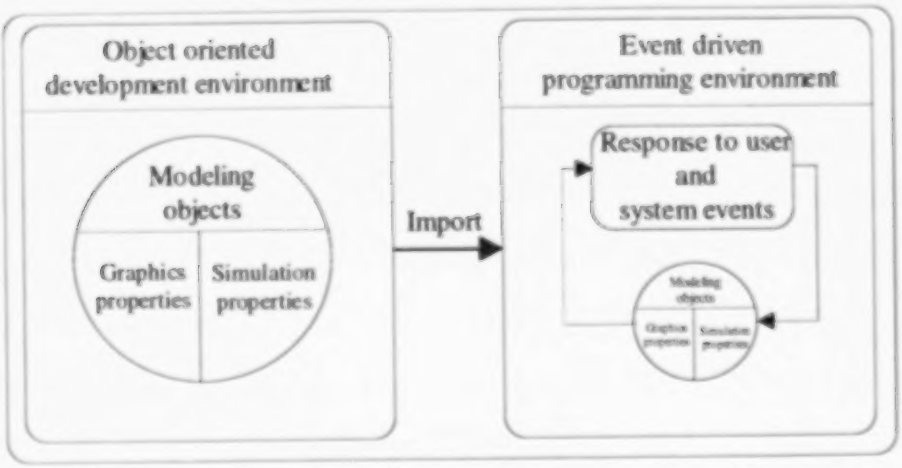


Figure 3-2: Implementation strategy for the model definition module

As such the implementation of HSM involved accomplishment of two tasks. First, the modeling objects were implemented in Visual C++. This task required identification of the graphic and simulation requirements for each object. Second, a graphical user interface was developed by using these modeling objects in Visual Basic. In the following sub-sections an outline of these two tasks is provided. A description of the internal structure of the translation module is described in the last sub-section.

3.4.1 Implementation of HSM modeling elements

Table 3-1 provides a list of the HSM modeling elements, a brief description and a graphical representation.

Table 3-1: HSM modeling elements

Element	*Graphical format*	*Description*
OPERATION		Used in the project WBS to define operations.
PROCESS		Used in the project WBS to define processes.
ALLOCATE RESOURCE		Used in the process models to allocate resources. This element is a special type of QUE.
FREE RESOURCE		Used in the process models to free the allocated resources. This element is a special type of FUNCTION.
PREDECESSOR		Used in the process models to define process interdependency with one or more successor processes. This element is a special type of FUNCTION.
SUCCESSOR		Used in the process models to define a dependency on a predecessor process. This element is a special type of QUE.
NORMAL		Used in the process models to define a non-constraint work task.
COMBI		Used in the process models to define a work task constrained by the availability of more than one type of resource.
QUE		Used in the process models to define a node where idle resources wait.
FUNCTION		Used in the process models to define new entities.
COUNTER		Used in the process models to keep track of the number of times a unit passes it.

The elements described in Table 3-1 were implemented as "classes" in C++. A class in C++ is a user defined type that defines "data members" and "member functions" for the class. An instance of a class is called an "object". Each object inherits the structure defined for the class through a property called "inheritance" (Microsoft, 1993b).

Normally, a program can be built in C++ (or any other object oriented programming environment) by first defining classes and then utilizing these classes in the program. In the implementation of HSM, however, a different approach was followed as previously illustrated in Figure 3-2. The HSM elements were programmed as classes (in this chapter the word "object" is used to refer to a C++ class and the word instance to refer to a C++ "object") in C++ such that the "data members" and "member functions" could communicate with the program written in Visual Basic.

Under this strategy each object is composed of two main components: an object model and an object procedure. The object model is a C-language structure that determines some fundamental attributes of the object with respect to its graphical behavior and simulation properties. Two of the most important fields within the object model are pointers to the following tables:

- The property information table, which lists all properties of the object along with its attributes.
- The event information table, which lists all the events applicable to the object along with information about arguments.

The object procedure is analogous to a window procedure in a Windows application. Its main function is to receive messages and respond to them. One of the most important things an object procedure can do is to determine when an event is recognized. Figure 3-3 shows a schematic representation of the internal structure of the modeling object. The modeling objects have various graphics and simulation properties along with message handling procedures as illustrated in Figure 3-3.

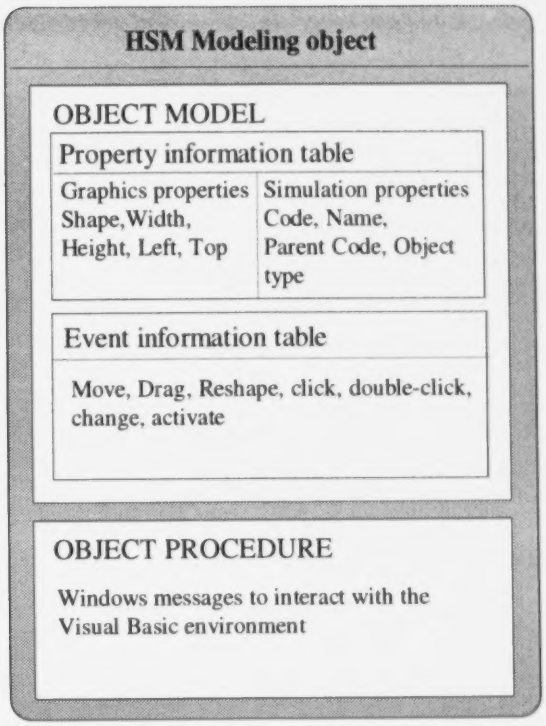


Figure 3-3: Schematic representation of the modeling object

3.4.2 Creation of Objects in C++

The above described generic structure of the objects was implemented in Visual C++. Apart from the requirements of developing objects, the code included some specific features to interact with Visual Basic. Figure 3-4 shows the C++ code for the QUE element. The code included in the Figure pertains to the property definition table, event definition table and object model for this object. As shown in the Figure the properties and events listed include both "graphic" and "simulation" properties. For example, the "top" and "left" properties, which are "graphic" properties, are utilized in positioning the object on the screen during a session of HSM. Similarly the "ProcessID" property, which

is a "simulation" property assists in determining the process model to which the element belongs.

```
// Property list
PPROPINFO QUE_Properties[] =
   {
   PPROPINFO_STD_LEFT,
   PPROPINFO_STD_TOP,
   PPROPINFO_STD_WIDTH,
   PPROPINFO_STD_HEIGHT,
   PPROPINFO_STD_VISIBLE,
   PPROPINFO_STD_TAG,
   &Property_QueId,
   &Property_ProcessId,
   &Property_Follower,
    NULL
   };
//---------------------------------------------------------------------------
// Event list
PEVENTINFO QUE_Events[ ] =
   {
   PEVENTINFO_STD_CLICK,
   PEVENTINFO_STD_DRAGDROP,
   PEVENTINFO_STD_GOTFOCUS,
   PEVENTINFO_STD_KEYPRESS,
   PEVENTINFO_STD_KEYUP,
   NULL
   };
//---------------------------------------------------------------------------
// Model struct
MODEL modelQUE =
   {
   VB_VERSION,                          // VB version being used
   0,                                   // MODEL flags
   (PCTLPROC)QueCtlProc,                // Control procedure
   CS_VREDRAW | CS_HREDRAW,                     // Class style
   sizeof(QUE),                                 // Size of QUELE structure
   IDBMP_QUE,                   // Palette bitmap ID
};
```

Figure 3-4: C++ code for object model of the QUE element

Figure 3-5 provides a partial listing of the object procedure developed to provide graphics capabilities to the object. Most of the object procedure is related to interaction between the object and the Visual Basic program. The control procedure shown is for the

QUE element and includes the listing of the "paint" procedure that creates the graphical representation of the QUE as shown earlier in Table 3-1.

```
//-------------------------------------------------------------------------
// QUE Control Procedure
{
        switch (msg)
        {
        case WM_NCCREATE:
        {
        LPQUE lpque = LpqueDEREF(hctl);
        lpcirc->QueName[] = "Queue1";
        lpcirc->QueId = 1;
        lpcirc->Follower = 1;
        break;
        }
        case WM_PAINT:
                if (wp)
                        PaintQ(hctl, hwnd, (HDC)wp);
                else
                {
                        PAINTSTRUCT ps;
                        BeginPaint(hwnd, &ps);
                        PaintQ(hctl, hwnd, ps.hdc);
                        EndPaint(hwnd, &ps);
                }
                break;
// Paint routine for displaying the queue
VOID NEAR PaintQ(HCTL hctl, HWND hwnd,HDC hdc)
{
        RECT rect;
        LPSTR  lpstr;
        LPQUE lpque = LpqueDEREF(hctl);
        HFONT  hfontOld = NULL;
        GetClientRect( hwnd, &rect);
        Ellipse(hdc, rect.left, rect.top, rect.right, rect.bottom);
        MoveTo(hdc, (rect.left+rect.right)/2, (rect.top+rect.bottom)/2);
        LineTo(hdc, rect.right, rect.bottom);
        lpstr = VBDerefHsz(lpque->hszCaption);
  DrawText(hdc, lpstr, -1, &rect, DT_VCENTER | DT_CENTER | DT_SINGLELINE);
}
```

Figure 3-5: Object procedure for the QUE element

The programming of all the elements follows the algorithm described above. The remaining details of the C++ implementation are provided in Appendix A. Appendix A

also provides the Visual Basic program listing and data structure for the storage of project information.

3.4.3 Programming in Visual Basic with imported objects

The objects developed in Visual C++ were compiled into special files called "custom control" files. Custom controls are a special types of objects that can be imported and used in the Visual Basic programming environment. Once these objects are imported in Visual Basic as custom controls they can be utilized by an application developer in any programming implementation.

The behavior of imported objects is controlled in Visual Basic by a special type of message processing as illustrated in Figure 3-6. When a "custom control" is loaded and used in Visual Basic, Windows sends the control a message as illustrated in Figure 3-6. Visual Basic intercepts messages intended for the control and forwards the message to the control. The message is then handled and processed by the default control procedure. Visual Basic appends a "control handle" to this message before sending it to the custom control. The control handle is basically a mechanism that allows access to the object model and object procedure of the custom control. For example, a user loads an instance of the QUE object while using HSM and provides the "QUE ID" (a property defined in the property table of the QUE object). This initiates a message that is sent to the custom control with a control handle appended to it. This message then accesses the object model of the custom control and updates the specified property. This is the general procedure through which the properties of the custom control are manipulated and forms the basis of program development using custom controls in Visual basic. The programming effort required development of procedures that could perform the following tasks:

1. instantiation of the modeling object for a project.
2. solicitation of the information pertaining to the instantiated objects.
3. provision of interactive features for the instantiated objects.

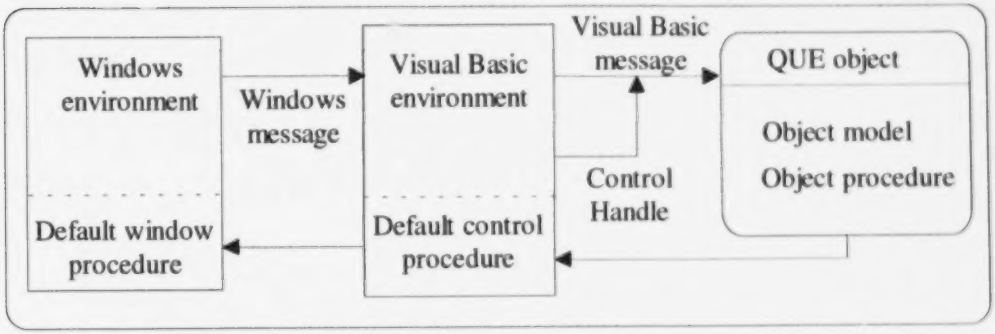


Figure 3-6: Message processing in Visual Basic

The first task in the Visual Basic part of the implementation was to write a code that would allow the user to instantiate the modeling objects for a project during a session of HSM. For example during the definition of the WBS the user should be allowed to graphically attach operations and process to the project hierarchy. This was achieved by allowing the user to create multiple instances of these modeling objects. Figure 3-7 shows a portion of the code written in Visual Basic for such task.

```
Sub Form_MouseDown (Button As Integer, Shift As Integer, X As Single, Y As Single)

If AddOpr% <> 1 And AddProcess% <> 1 Then Exit Sub
   ' instantiating an operation
     Load Opr1(NOpr%)
     Opr1(NOpr%).Top = Y
     Opr1(NOpr%).Left = X
     Opr1(NOpr%).Visible = True
     Opr1(NOpr%).Caption = OprEdit.Text1.Text
     Opr1(NOpr%).Tag = Str$(OprEdit.Option3D1.Value)
     Opr1(NOpr%).OprId = Val(OprEdit.MaskedEdit1.Text)
     Call ConnectOpr
```

Figure 3-7: Visual Basic procedure for creating instances of modeling objects

Figure 3-7 shows the subroutine that instantiates the "operation object". This subroutine is invoked when the user presses the left mouse button.

The second task in the programming required development of subroutines that would allow the user to provide information related to the properties of the modeling object. For example after instantiating the operation object the user is provided with simple input forms to solicit information like operation name, parent and code. This part of the program was greatly enhanced by the graphical features of Visual Basic and did not require special programming effort.

The third part of the programming process pertains to the interactive features provided in HSM. The objects that are instantiated by the user either in the project WBS or the process models are displayed on the screen. In order to enhance the user interface, procedures were written in Visual Basic that would provide the user with "drag-drop", "move" and "edit" features. This required manipulating the object procedures for the modeling object from within Visual Basic. Figure 3-8 illustrates the "move" procedure of the operation and process object.

```
Sub Form_DragDrop (Source As Control, X As Single, Y As Single)

If MoveWhat = 1 Then
   Move operation object
   Opr1(OprIndex%).Move X, Y
   Call DrawLine
   Call DrawLines
Elseif MoveWhat = 2 Then
  Move process object
   Process1(ProcessIndex%).Move X, Y
   Call DrawLine
   Call DrawLines
Elseif MoveWhat = 3 then
```

Figure 3-8: Subroutine for "move" event for the operation object

Apart from the above three tasks the implementation required programming of other procedures for file handling, error trapping and for translating the simulation models.

3.4.4 Description of the translation module

The function of the translation module is to translate the information provided in the model definition module into a single simulation model. The information provided in the project WBS, resource library, operation sequencing and process models is compiled by this module. In the current form HSM utilizes SLAMSYSTEM (Pritsker, 1992) as the general purpose simulation language. As such the translation is geared towards development of a single simulation model using the SLAMSYSTEM modeling concepts. The enhancements made to CYCLONE required use of a simulation engine that would support the new modeling elements. Keeping in mind the objective of this research it was decided not to enhance the MicroCYCLONE code or develop a new simulation engine as both are taxing approaches. This basically reduced the problem to selection of a general purpose simulation language. Choice of SLAMSYSTEM was influenced by the author's experience with the language and its proven performance.

Figure 3-9 illustrates the working of the translation module. The translation algorithm adopted for HSM is as follows:

- Compile the information from the WBS, operation sequencing and process interdependency to link all the processes in the project.
- Utilize the information in the resource library to initialize the resources in "resource blocks" (a SLAMSYSTEM node). Based on the information provided for resource allocation formulate a priority list for the common resources.

- Translate individual process models into equivalent SLAMSYSTEM models. This task includes duplication of the modular processes.

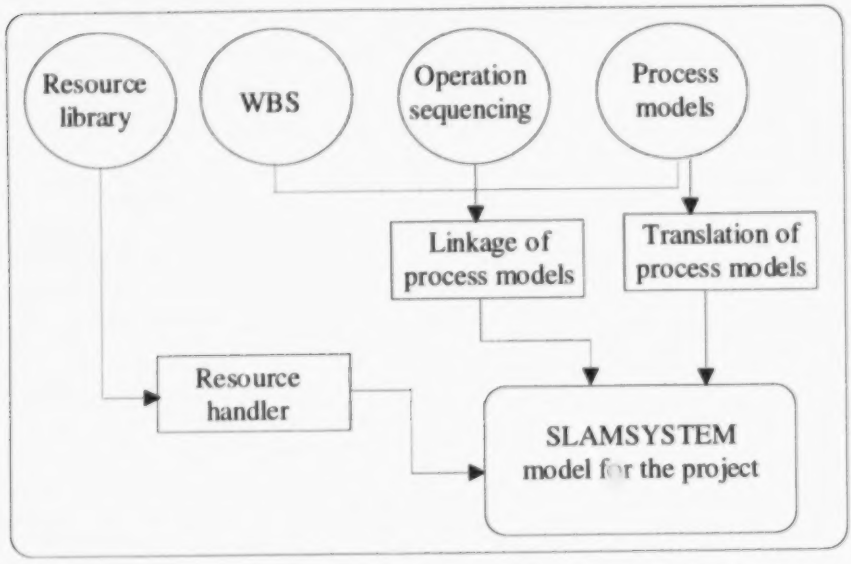


Figure 3-9: Internal working of the translation module

Appendix B provides the implementation details of the translation module.

3.5 SAMPLE SESSION

The operation of HSM is centered around four major sessions which are:

- Work breakdown structuring.
- Resource management.
- Operation sequencing.
- Process modeling.

WBS session: This session provides all features for the development of the project WBS. Figure 3-10 shows the WBS screen which consists of a "pull-down" menu bar and a drawing area. A resource manager icon and project element are automatically generated for each new project.

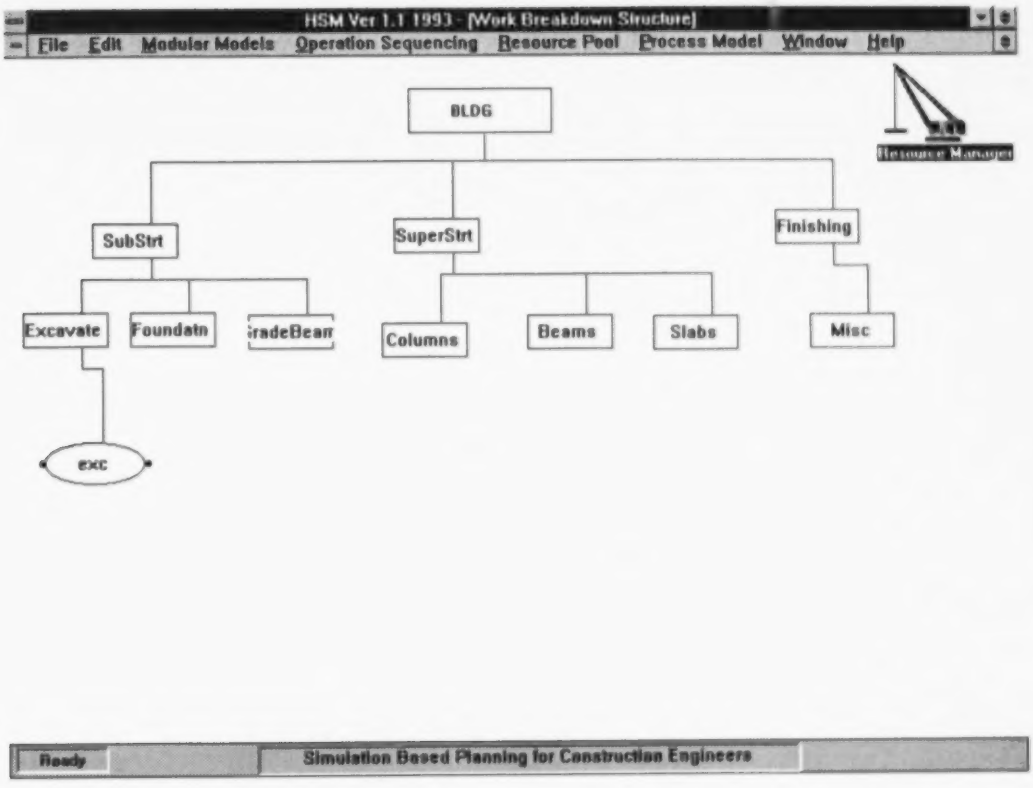


Figure 3-10: Actual WBS screen

The WBS screen acts as the template upon which the user is allowed to add, delete or edit operation and process elements using the input forms shown in Figure 3-11 and Figure 3-12.

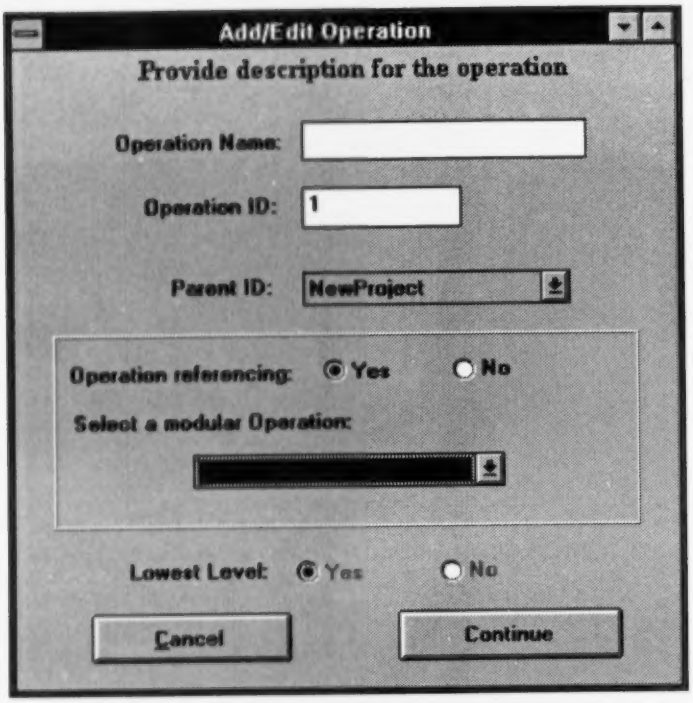


Figure 3-11: Input form for an operation element

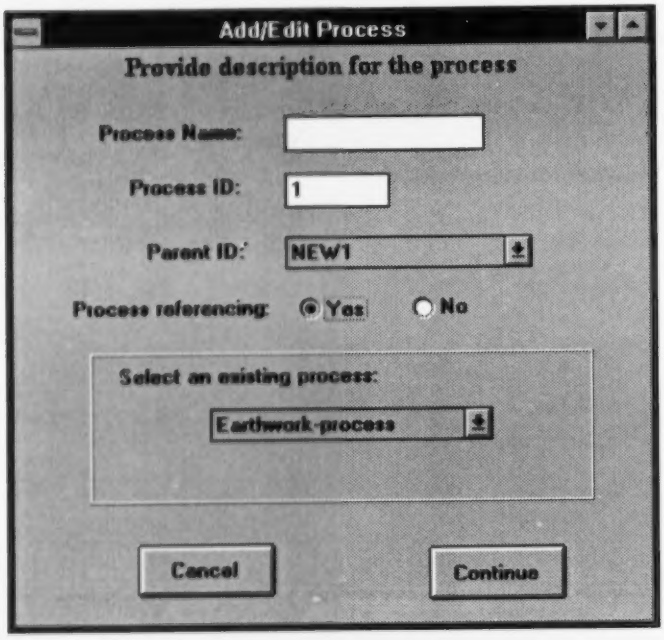


Figure 3-12: Input form for process element

Resource management: The resource manager acts as a tool that coordinates initialization of resources and other issues related to their management. Figure 3-13 shows the resource manager screen. The user invokes this screen by clicking on the resource manager icon on the WBS screen. HSM allows the user to add, edit and delete resources for a project. Simple input forms are provided to solicit information from the user. One such screen in which the user adds a resource to the project is shown in Figure 3-14.

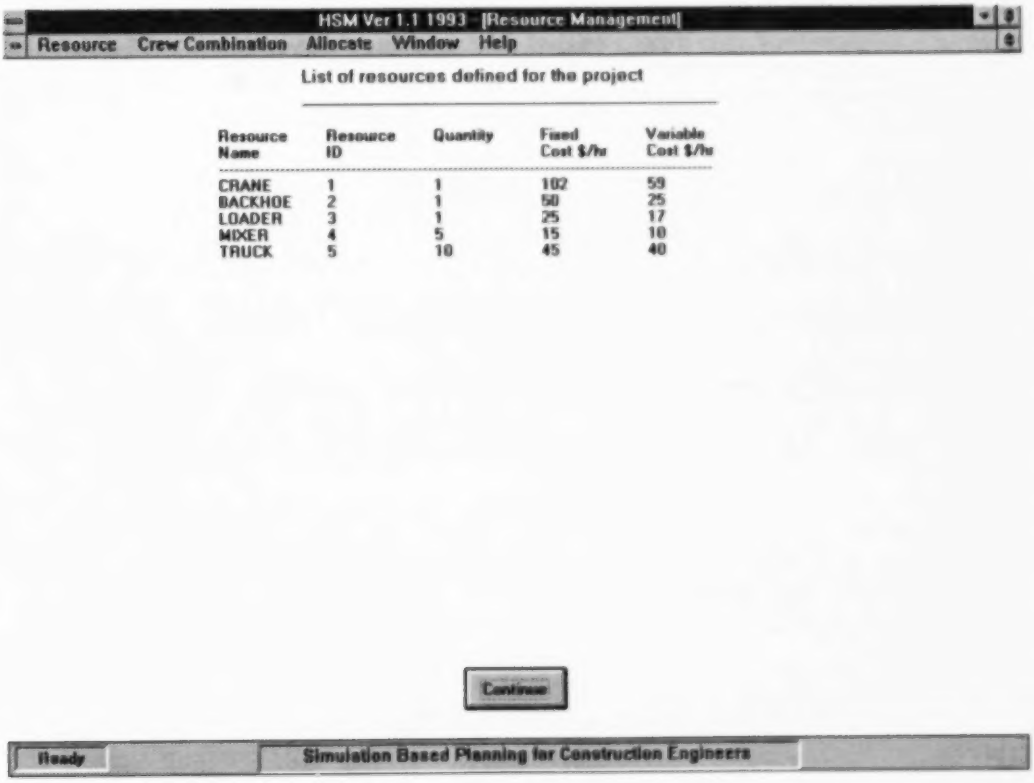


Figure 3-13: Resource manager screen

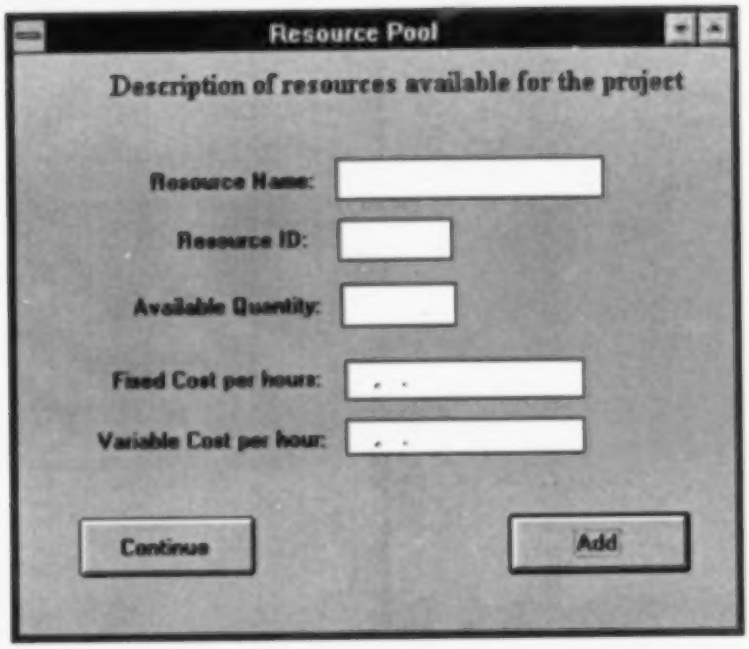


Figure 3-14: Input form for adding a resource

Operation sequencing: The operation sequencing session can be invoked by the user after definition of the project WBS. This session allows the user to define links between operations that exist in the project WBS. These links are graphically displayed in the drawing area of the screen. The user is provided simple features that include adding, editing and deleting links. Figure 3-15 shows an operation sequencing screen.

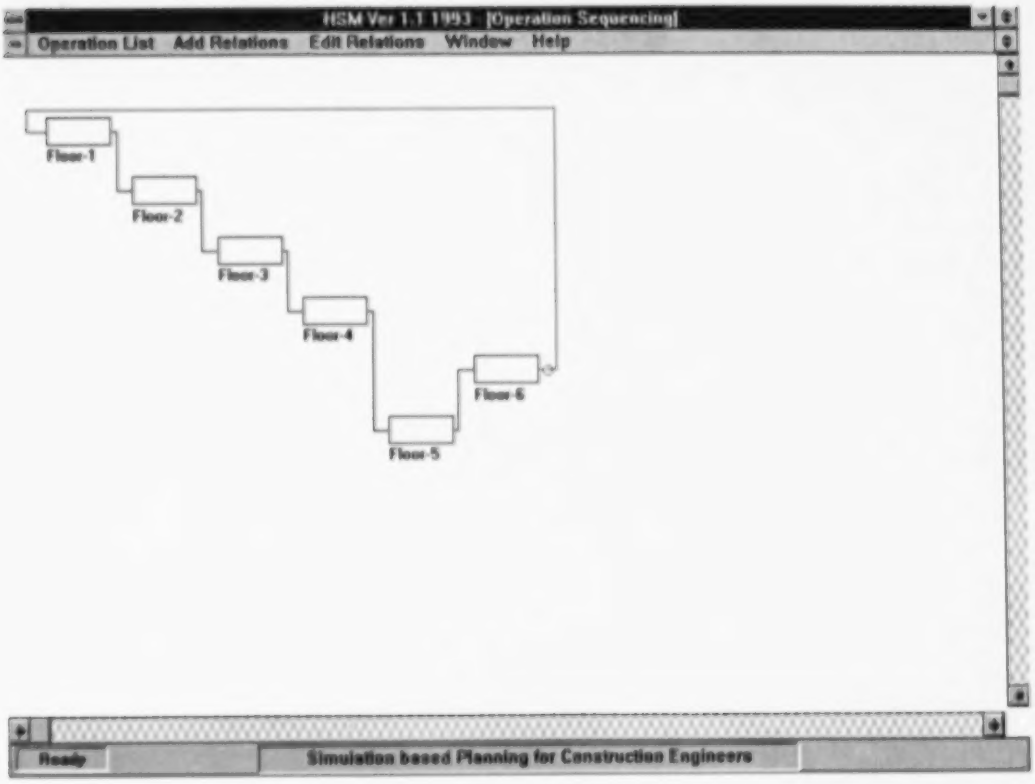


Figure 3-15: Operation sequencing screen

Process modeling: The process modeling session acts as a template upon which the user is allowed to graphically develop the process models. In addition to the features provided on the other screens this screen includes a tool bar on which the modeling objects are displayed. For developing a process model the user can use the "drag-drop" feature to paste any modeling object on the drawing area. Figure 3-16 shows the process modeling screen with modeling objects pasted by the user on the drawing area. Once the modeling object is pasted on the drawing area the user is provided with an input form on which the information for that modeling object (element) is provided. Figure 3-17 shows the input screen for a COMBI element.

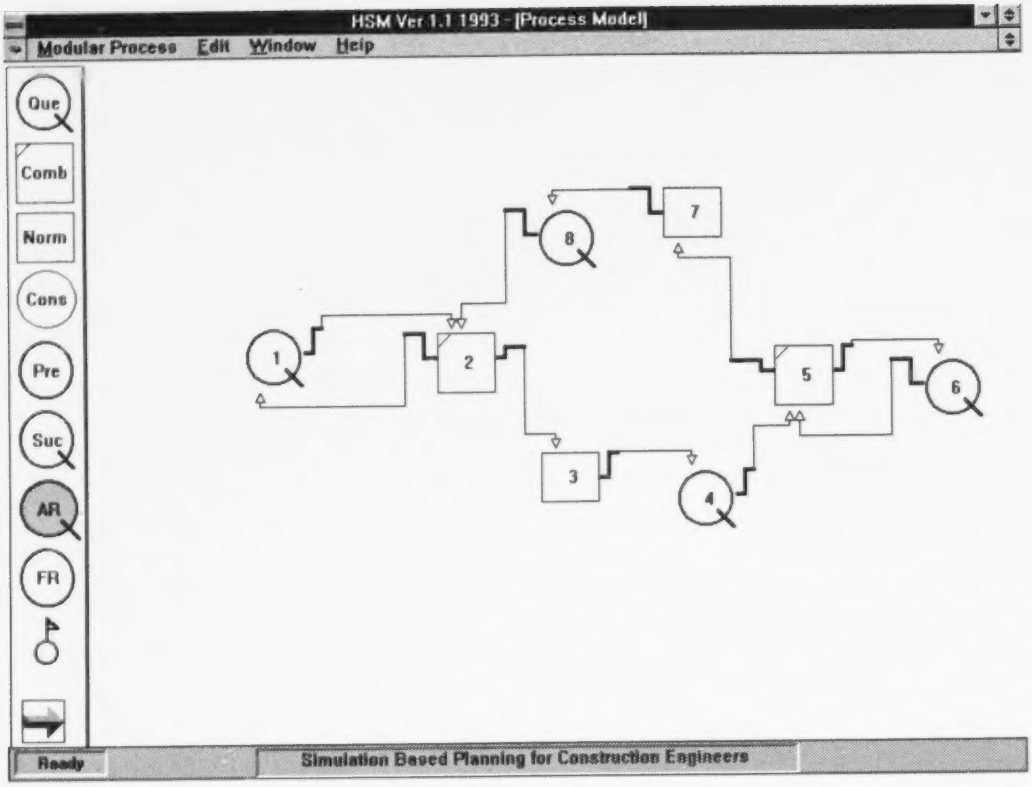


Figure 3-16: Process modeling screen

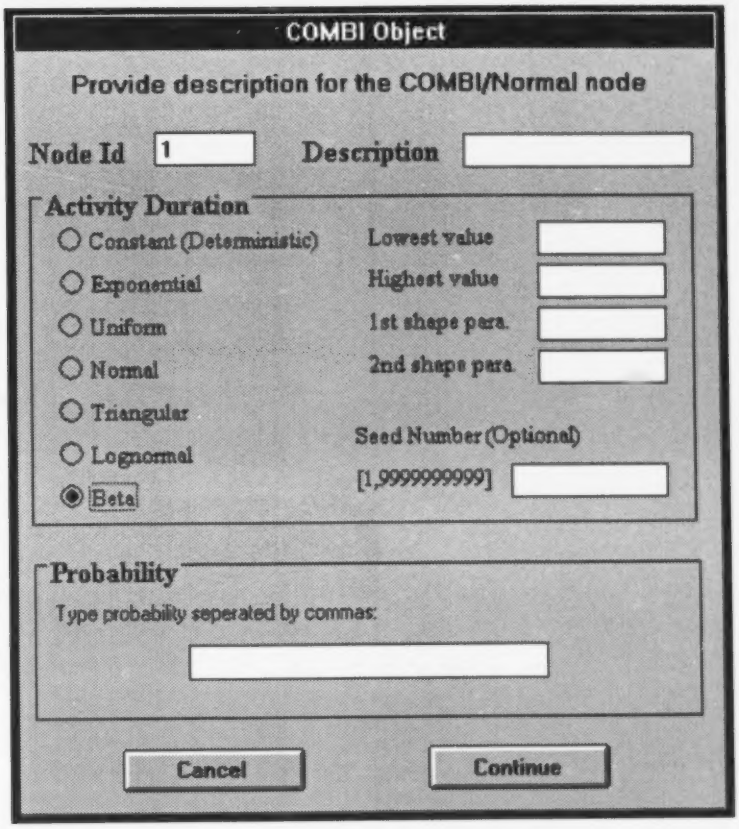


Figure 3-17: Input form for COMBI element

3.6 LIMITATIONS OF THE PROTOTYPE SYSTEM

The HSM prototype was developed to demonstrate the benefits of simulation based planning and to test and validate the HSM modeling concepts. In the current form the prototype has some limitations. The program utilizes a general purpose simulation system as its simulator and as such it does not provide a seamless interface between the four modules. The program is not "intelligent" to detect mistakes in the simulation models that the user develops and as such does not detect redundant information. In future versions of the program it is suggested that an internal simulation engine should be developed. This

will make the program operations automated so that the user will only be required to interact with the model definition module and the report generation module.

3.7 CONCLUDING REMARKS

This chapter demonstrated the use of object oriented concepts to enhance simulation modeling by improving visual modeling and by allowing hierarchical and modular constructs. The use of modeling objects that have both "graphic" and "simulation" properties benefit simulation modeling in general and construction simulation in particular. Object oriented simulation can be effectively applied in construction to reduce the cognitive gap between the simulation models and the real world problems. This could lead to increased popularity of simulation and simulation based tools in the construction industry.

Chapter 4: Planning a Bridge Project using HSM

4.1 INTRODUCTION

The completed Peace River bridge project was used as a case study in this research. This chapter describes planning of the sub-structure of this project using HSM. The objectives were:

1. To test and validate the HSM modeling concepts.
2. To demonstrate the feasibility of project level simulation.
3. To illustrate the potential benefits of simulation-based planning for construction projects.

A general description of the bridge project and its pertinent construction details are provided. A step by step description of the plan development and subsequent experimentation with the developed simulation models is also discussed.

4.2 GENERAL DESCRIPTION OF THE PEACE RIVER BRIDGE

The bridge across the Peace River 18 km northeast of Weberville in the province of Alberta, was contracted by the Alberta Transportation and Utilities (ATU), using the following contracting methods:

1. Contract 1: The construction of the sub-structure of the bridge.
2. Contract 2: Off-site fabrication of the steel girders, delivery, erection and final positioning of the girders constituted the second contract.
3. Contract 3: Construction of the reinforced concrete deck, approaches and other miscellaneous work.

The construction of the bridge started in 1989 and was completed in 1991. Table 4-1 provides an approximate quantity estimate for the bridge (ATU, 1989).

Table 4-1: Quantity estimate for Peace River Bridge (ATU, 1989)

Item	Unit	Sub-structure	Super-structure
Compacted fill and grading west end	Lump sum	-	-
Excavation and backfill east end	Lump sum	-	-
Excavation - structural	Lump sum	-	
Backfill - compacted granular	Lump sum	-	-
Steel H piling (HP 310 x 94)			
- Set-up	m	152	-
- Splice	splice	256	-
- Drive	pile	3,358	-
Pre-drill hole for H-piling - abutments	Lump sum	-	-
Concrete - Class A (Pier-1)	m^3	792	-
Concrete - Class C (Abutments)	m^3	751	-
Concrete - Class C (Pier shafts)	m^3	5,888	-
Concrete - Class C (Deck)	m^3	-	2,740
Reinforcing Steel - Plain	kg	700,509	159,955
Reinforcing Steel - Epoxy coated	kg	22,066	340,137
Structural steel	t	-	4,376
Bridge rail	m	-	1,537
Miscellaneous iron	Lump sum	-	-
Polymer waterproofing membrane	m^2	311	8,166
Asphaltic Concrete Wearing Surface	m^2	425	8,166

The bridge has seven spans, 5 spans of 112 m, 1 span of 82 m and 1 span of 92 m, with a total length of 734 m. The bridge has a grade of 0.7% sloping down from the west bank towards the east bank of the river. Figure 4-1 shows a schematic representation of the bridge.

The specifications for the contract provided two options for the construction of the substructure. The first option involved for each pier a spread footing foundation while the second option used concrete caissons for the pier foundations. The bridge was constructed using the first option.

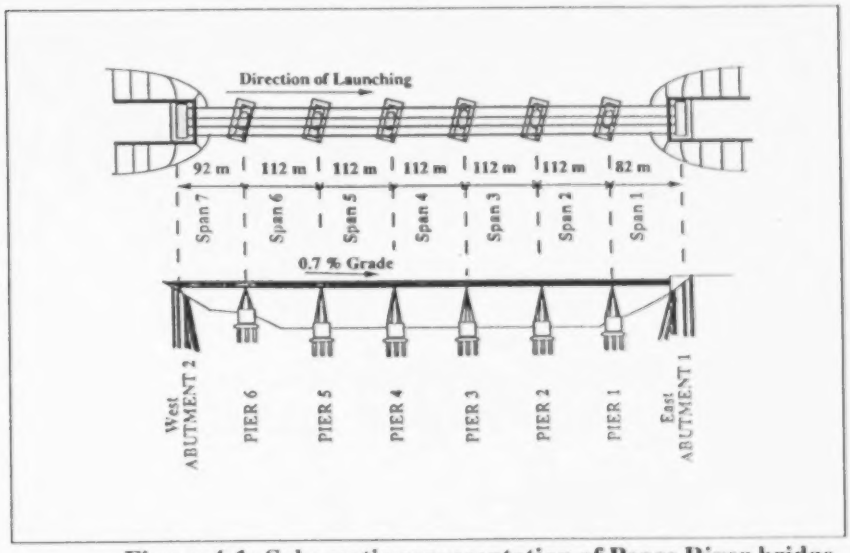


Figure 4-1: Schematic representation of Peace River bridge

Figure 4-2 shows a schematic representation of a typical bridge pier. The dimension of the piers varies based on their location. Table 4-2 lists the approximate quantity estimate for a typical pier (ATU, 1989).

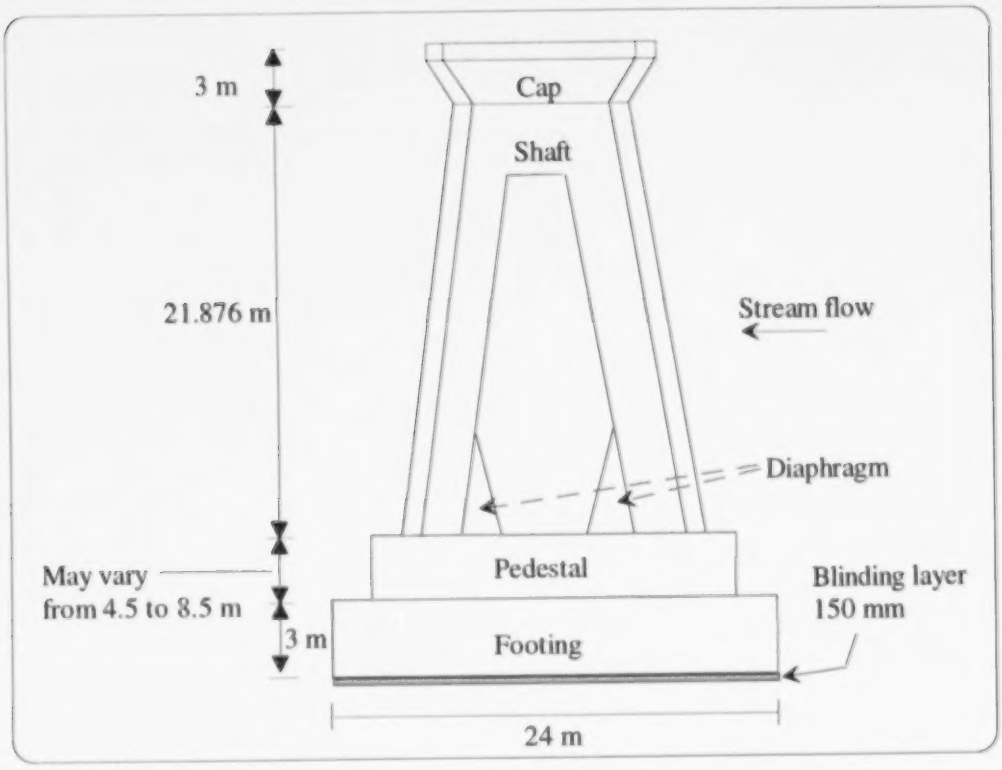


Figure 4-2: Schematic representation of a typical pier

Table 4-2: Quantities for the typical bridge pier (ATU, 1989)

ITEM	UNIT	ESTIMATE
Class 'S' concrete blinding layer	m^3	43
Class 'A' concrete footing	m^3	90
Class 'A' concrete pedestal	m^3	1,400
Class 'A' concrete shaft and cap	m^3	1,098
Reinforcing steel (footing and pedestal)	kg	97,667
Reinforcing steel (shaft and cap)	kg	109,298

Pier-1 of the bridge was constructed on the east shore and constituted of the installation of steel piles, blinding layer, pier footing, pier shaft and diaphragm, and pier

cap. Since Pier-1 was constructed on the shore hence no pedestal was required. All of the remaining piers had pedestals of varying heights. The excavation process used a hydraulic excavator. Upon completion of the excavation, steel piles were driven and the blinding layer was placed. Footing formwork, rebar and concreting processes were then performed sequentially. The pier shaft was constructed in two lifts and then the cap was constructed.

Pier-2 was accessible from the west bank and as such did not require the construction of a berm. Similarly Pier-6 was accessible from the east bank and did not require the construction of a berm. The construction details for Pier-2 to 6 were exactly the same. For Pier-3 a berm was created from the west bank whereas for Pier-5 and Pier-4 the berm was constructed from the east bank.

Piers 2 to 6 are typical piers and were constructed by excavating in the river to the top of the shale surface, placing the blinding layer, pouring the concrete footing, pedestal (with varying height depending on the depth of the river), pier shaft and diaphragm, and pier cap.

For the typical piers, a steel frame was utilized for constructing a cofferdam. The construction of the cofferdam was started after the construction of the respective berm was completed. The steel frame was floated in place, positioned and anchored. This was followed by driving of spuds and steel sheet piles. During the cofferdam piling activity for the cofferdam process continuous dewatering was essential. Upon completion of the cofferdam, excavation was performed using a clamshell. This was followed by the

placement of the blinding layer, and the remaining operations as described for Pier-1. The construction of Pier-2, 3, 4, 5 and 6 required casting of pedestals of varying heights.

The two abutments and wing walls were cast on steel piles. The bridge deck is supported by four 4.545 m deep steel girders with diaphragms and lateral bracing. The girders were shop-fabricated, shipped to the site and erected by launching from the west bank. The bridge was launched using a launching nose and roller arrangement. The launched height of the girders was 800 mm higher than the final elevation. The girders were than jacked down to their final position. The reinforced concrete bridge deck had a polymer waterproofing membrane and an asphalt concrete wearing surface.

4.3 SCOPING THE BRIDGE PROJECT

The prototype program was used in scoping the plan for the Peace River Bridge Project. The focus of this research was on the first contract: the sub-structure part of the project. The first step was to develop the project WBS. Various operations and processes of the project were identified in a top-down fashion.

The graphical user interface of HSM greatly facilitated the development of the project WBS. The WBS for the sub-structure part is shown in Figure 4-3.

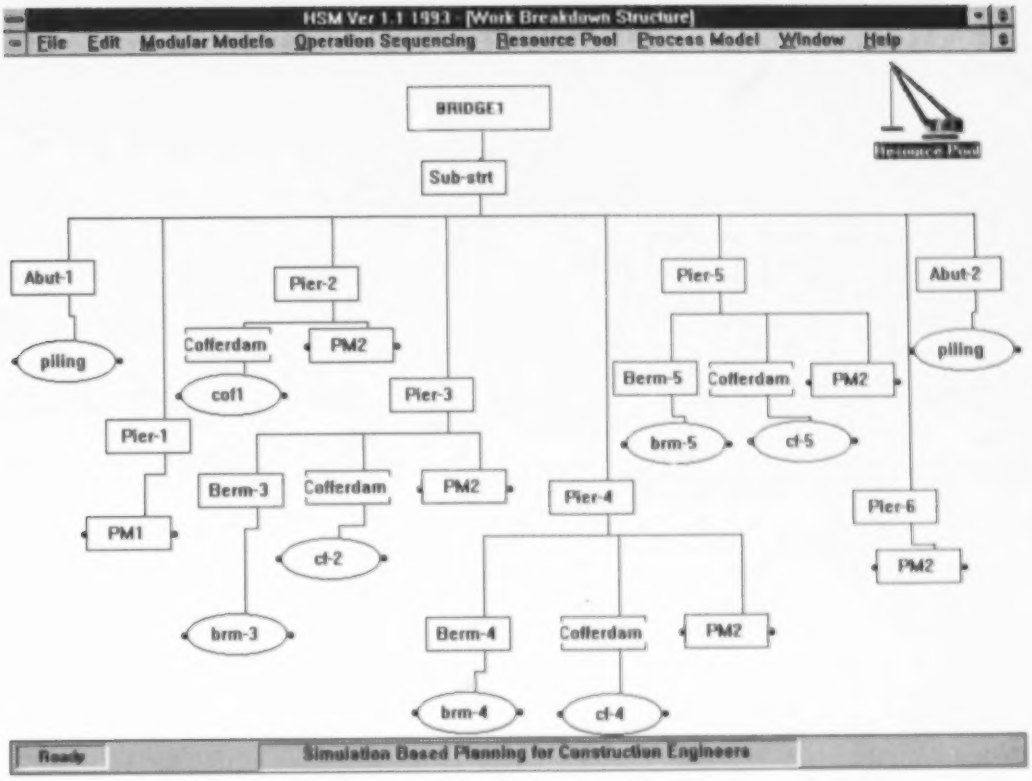


Figure 4-3: WBS for substructure of the bridge project

In the development of the WBS, the modularity feature of HSM was extensively utilized. At "level-1", construction of the two abutments and six piers were identified as operations. In the substructure phase of the bridge construction, the abutments were built to the lower level of the wing walls so as to allow launching of the girders. Therefore only the installation of steel piles and bridge seats was completed during the substructure-phase of the project. As such the Abutment-1 and Abutment-2 operations were broken down into the "piling" process. This is illustrated in the WBS by the piling processes attached to the two abutment operations. The piling process is a modular process that is referenced by both abutments (a modular process is denoted in Figure 4-3 by ellipse with two dots).

A generic piling process model was defined and was then used for Abutment-1 with 32 steel piles and for Abutment-2 with 30 steel piles. In order to simplify the WBS, two modular operations were defined for the project. The construction details of Pier-1 were defined in a modular operation "PM1" which is shown in Figure 4-4. This modular operation had "excavation", "piling", "blinding layer", "footing", "shaft" and "cap" operations were defined as level-1 operations. Corresponding processes were defined for the "excavation", "piling", "blinding layer", "footing" and "cap" operations. The "shaft" operation was further divided into "formwork", "rebar" and "concreting" operations which in turn had the corresponding processes attached to them.

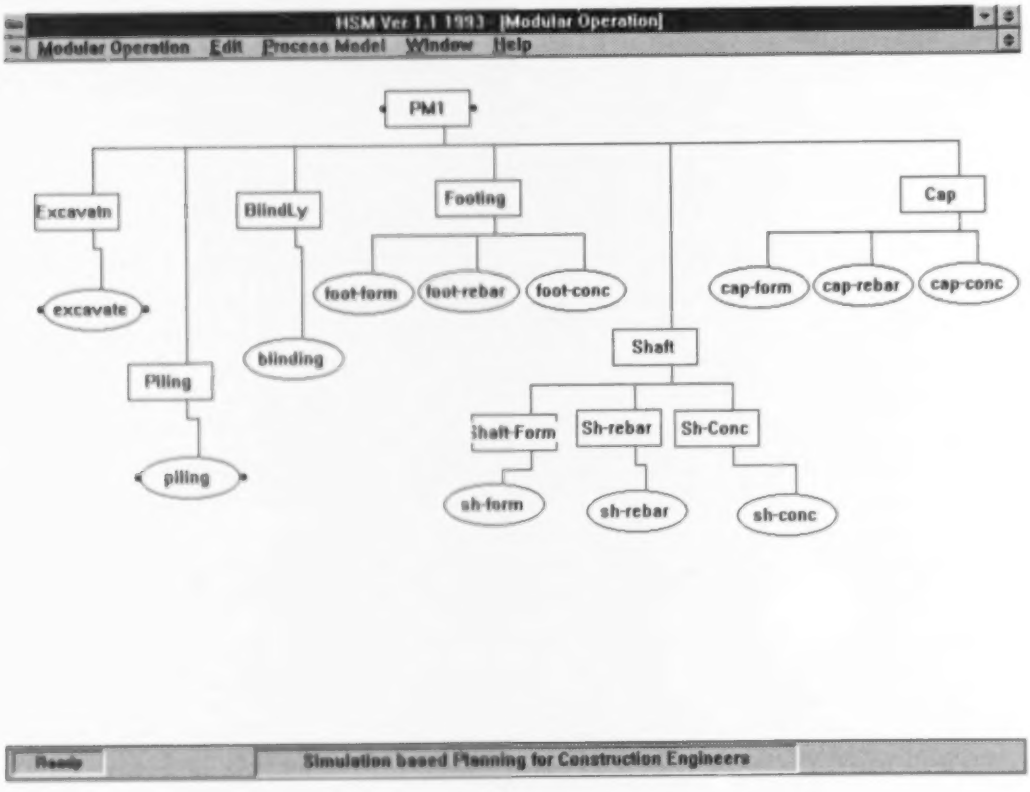


Figure 4-4: Modular operation for Pier-1

In the breakdown of the "Shaft" operation two options were available. Due to the height of the shaft it was necessary to construct it in two lifts. The first option available to model this situation required definition of six processes namely: "formwork first-lift", "rebar first-lift", "concreting first-lift", "formwork second-lift", "rebar second-lift" and "concreting second-lift". In the second option the same situation could be modeled by dividing the "Shaft" operation to another level of operations and then linking them in a cyclic fashion. The author used the second option for this study. The "Shaft" operation was first divided into three operations namely: "Shaft-formwork", "Shaft-rebar" and "Shaft-concreting". Then "formwork" process was attached to the "Shaft-formwork" operation, "rebar" process was attached to the "Shaft-rebar" operation and "concreting" process was attached to the "Shaft-concreting" operation. By using the cyclic link between the "Shaft-formwork", "Shaft-rebar" and "Shaft-concreting" operations the construction of the shaft in two lifts was modeled.

As the same construction method was adopted for Piers 2, 3, 4, 5 and 6 a modular operation called "PM2" was designed and was referenced by these piers. Figure 4-5 shows the "PM2" modular operation which has its own WBS. The modular operation "PM2" consists of level-1 operations that include "blinding-layer", "footing", "pedestal", "shaft" and "cap" operations. These operations were then divided into the respective processes. HSM, provides modularity at both the operation and process level. These modular operations are internally stored in a "library" and can be used in the planning of future projects. The shaft and pedestal of these piers were also constructed in two lifts.

This was again modeled by first introducing an extra level of operations that were then combined by cyclic links.

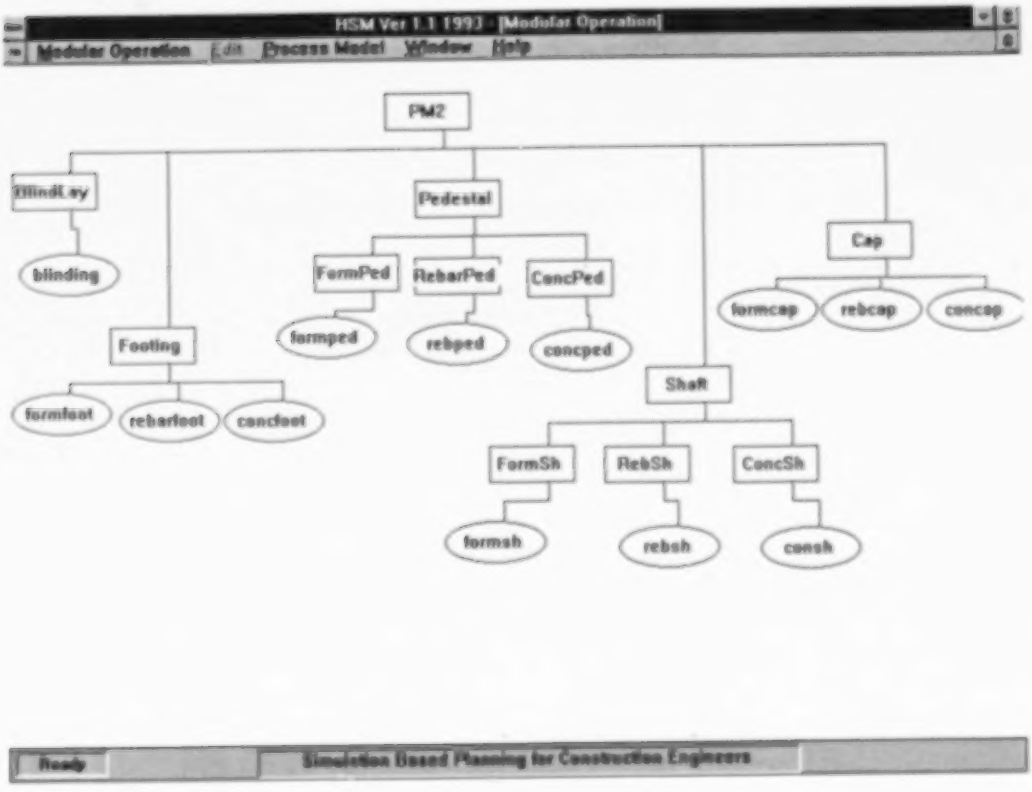


Figure 4-5: Modular operation for Pier 2, 3, 4, 5 and 6

Resource Definition

Resources required for the project were identified in the second step. Table 4-3 provides a list of resources that were allocated to the project. The table provides the quantity of each type of resource and the process numbers to which the resource is allocated in a decreasing order of priority. The priority for various processes is internally utilized by HSM during the simulation experiment in scheduling various work tasks that require common resources.

Table 4-3: Resources allocated to the bridge project

ID	Resource	Quantity	Processes
1	Crane 40t	1	1,4
2	Diesel hammer	1	1,4,41,83,85,87
3	Piling crew	1	1,4,41,83,85,87
4	Excavator west	1	3,2
5	Truck west	5	3,19
6	Concrete pump west	1	6,10,14,18,23,27,31,35,39,43,47,51,55,59
7	Transit mixer west	2	5,9,13,17,22,26,30,34,38,42,46,50,54,58
8	Concrete crew west	1	6,10,14,18,23,27,31,35,39,43,47,51,55,59
9	Footing form west	1	7,24,44
10	Crane 20t west	1	7,11,15,24,28,32,36,44,48,52,56
11	Form crew west	1	7,11,15,24,28,32,36,44,48,52,56
12	Crane 100t west	1	8,12,16,25,29,33,37,40,45,49,53,57,82
13	Rebar crew west	1	8,12,16,25,29,33,37,45,49,53,57
14	Shaft form west	1	11,32,52
15	Cap form west	1	15,36,56
16	Dozer west	1	20
17	Cofferdam steel cage	4	40,82,84,86
18	Pedestal form west	1	28,48
19	Truck east	5	78,80
20	Concrete pump east	1	61,65,69,73,77,89,93,97,101,105,107,111,115,119,123
21	Transit mixer east	2	60,64,68,72,76,88,92,96,100,104,106,110,114,118,122
22	Concrete crew east	1	61,65,69,73,77,89,93,97,101,105,107,111,115,119,123
23	Footing form east	1	62,90,108
24	Crane 20t east	1	62,66,70,74,90,94,98,102,108,112,116,120
25	Form crew east	1	62,66,70,74,90,94,98,102,108,112,116,120
26	Crane 100t east	1	63,67,71,75,84,86,91,99,103,109,113,117,121
27	Rebar crew east	1	63,67,71,75,91,95,99,103,109,113,117,121
28	Shaft form east	1	70,98,116
29	Cap form east	1	74,102,120
30	Dozer east	1	79,81
31	Pedestal form east	1	66,94,112

The process numbers shown in the table are not provided by the user, these are internally obtained by the program after the modeler has completed the resource allocation. The list of processes defined for the Peace River bridge are provided in Table C-1 in Appendix C.

In Table 4-3, the defined resources can be divided as follows:

1. The first group of resources are located on the east bank of the river. These resources are utilized in the construction of Abutment-2, Pier 4, 5, and 6.
2. The second group of resources are located on the west bank of the river. These resources are utilized in the construction of Abutment-1, Pier 1, 2 and 3.
3. The third group of resources are assumed to be mobile such that they can perform work either on the east or the west bank.

The above division of resources was optional and was defined keeping in mind the requirements of the construction site.

4.4 OPERATION SEQUENCING FOR THE BRIDGE PROJECT

The next step in the plan development involved a description of the implementation strategy for the example project. Figure 4-6 illustrates the procedure adopted in the prototype for performing this function by showing the sequencing of level-1 operations.

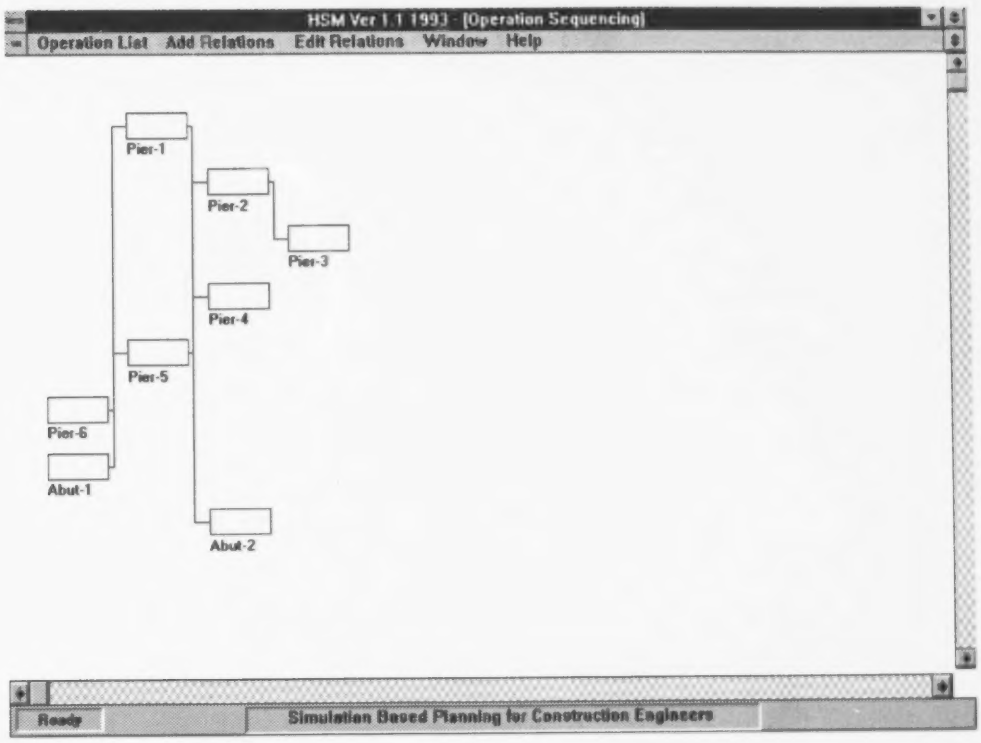


Figure 4-6: Operation sequencing using the prototype

The program allows the user to add links between the operations that are defined for the project. Using this facility the operation sequencing at all levels of the example project were defined. Figure 4-7, 4-8, 4-9 and 4-10 show the operation sequencing for level 1, 2, and 3 respectively. In Figure 4-7 the operation sequencing at level-1 of the project is shown. The operations are separated into two groups one being the piers constructed from the west bank and the other being the piers constructed from the east bank. On the west bank work was started at Abutment-1 and Pier-3 simultaneously. This is represented by a parallel link between these two operations. After the completion of Abutment-1, construction of Pier-1 was started which was then followed by Pier-2 and Abutment-2 in parallel. Pier-1 operation is therefore serially linked to Pier-2 and Abutment-2 operations.

The construction on the east bank was started at Pier-5 which was followed by Pier-4 and 6. Therefore Pier-5 is serially linked to Pier-4 and Pier-6. The Abutment-2 operation began after the completion of Pier-1 operation.

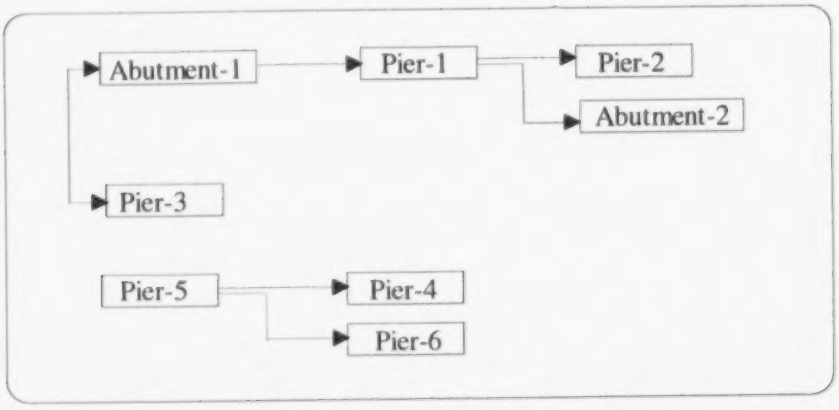


Figure 4-7: Operation sequencing at level-1

Figure 4-8 shows the sequencing of level-2 for Pier-1. All level-2 operations in this case are linked in a serial fashion as shown. The construction on Pier-1 started with the excavation operation and was completed when the concreting operation in the pier cap was completed.

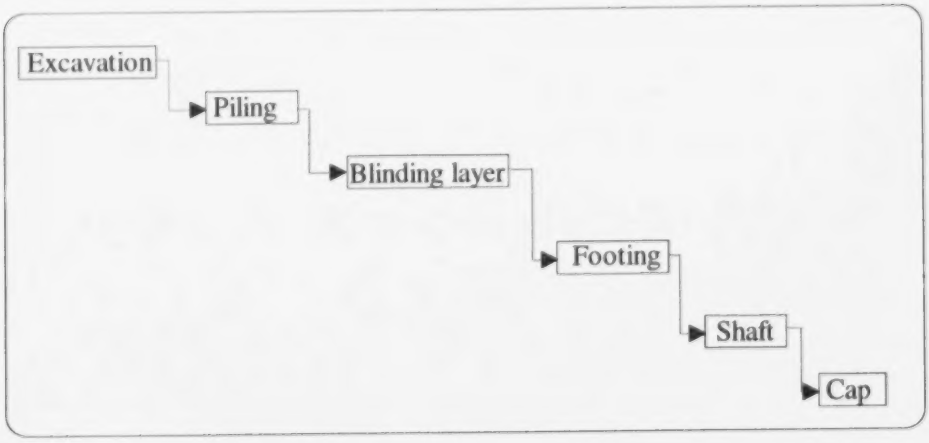


Figure 4-8: Operation sequencing for Pier-1 at level-2

Similarly Figure 4-9 shows the operation sequencing for the level-2 operations for Piers 2, 3, 4, 5 and 6. There is a serial link between level-2 operations. The construction of these piers was started by the blinding layer operation while the concreting of the cap marks the completion of each pier. During the actual plan development, the modeler was required to provide this sequencing individually for all five piers.

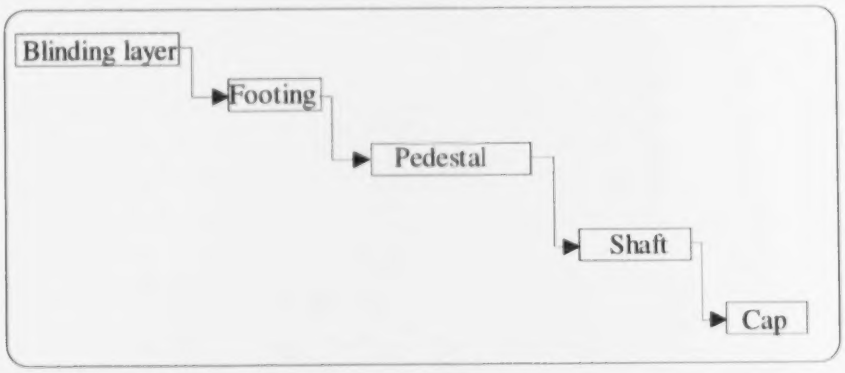
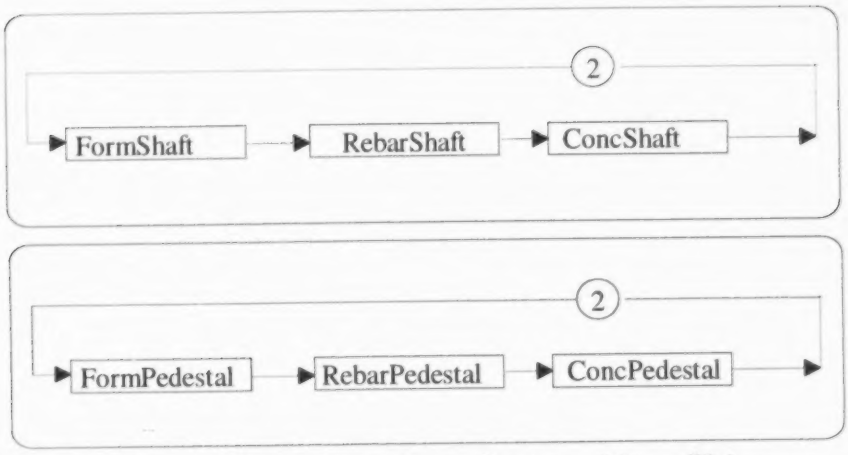


Figure 4-9: Operation sequencing for Piers 2, 3, 4, 5 & 6 at level -2

Figure 4-10 shows the operation sequencing of the operations at level-3. Sequencing at this level was provided for all the piers separately, only one figure is shown due to the similarity in sequencing. The shaft and the pedestal were both constructed in two vertical lifts. This situation was modeled in HSM by using the cyclic link between the "formwork", "rebar" and "concrete" for the shaft operations with a cyclic counter of two. The "formwork", "rebar" and "concrete" operations are first linked serially to denote the first lift and are cycled again to denote the work done during the second lift. The same sequencing was adopted for all the shafts and pedestals of the piers except Pier-1 which does not have a pedestal.



Level-3 (Sequencing using cycling to model two lifts)

Figure 4-10: Operation sequencing for Piers 2 to 6 at level-3

4.5 PROCESS MODELS FOR THE BRIDGE PROJECT

The next step in the plan development involved defining all of the process models identified for the example project. This exercise was also performed using the prototype. Presently the graphical user interface for the development of process models in the prototype is a "middle of the road" tool. It provides a very simple interface with limited on-screen display features. All the process models were developed using the HSM prototype. Due to the above mentioned limitation of the prototype actual screens are not used here.

Process models were developed for all the processes that were identified in the project WBS. Some of these process models were modular and were referenced by the corresponding processes. Figures 4-11 through 4-18 show the typical process models developed for the bridge project.

In Figure 4-11 the process model for the "berm" process is shown. Two resources are required to perform this process which include a dozer and five trucks. The trucks are

allocated to the berm process at an Allocate Resource Node and are modeled to arrive at the site every 15 minutes. The use of five trucks and 15 minute arrival rate was based on the productivity factors the contractor had from previous similar projects. The trucks dump their dirt load directly into the river and exit the site. Four truck loads are allowed to accumulate before the dozer starts pushing the dirt and shaping the berm.

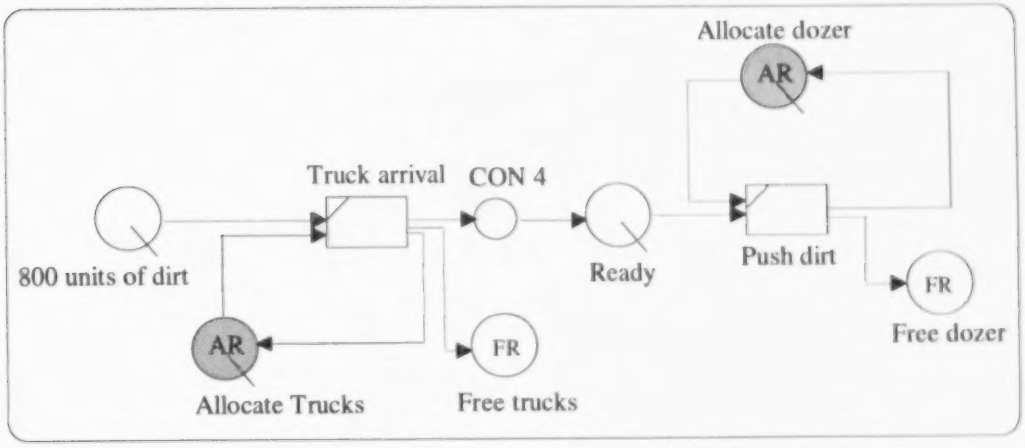


Figure 4-11: Process model for the berm process

Figure 4-12 shows the process model for the excavation process for Pier-1. In this process one excavator and ten dump trucks are utilized. Since the excavation process had to be completed before work on Pier-1 could proceed further, the Allocate Resource Nodes for both the trucks and excavator were arranged in such a manner that they are released after the excavation was completed. This was achieved by placing the Allocate Resource Node before two dummy COMBI's and placing the Free Resource Nodes after the CONSOLIDATE function.

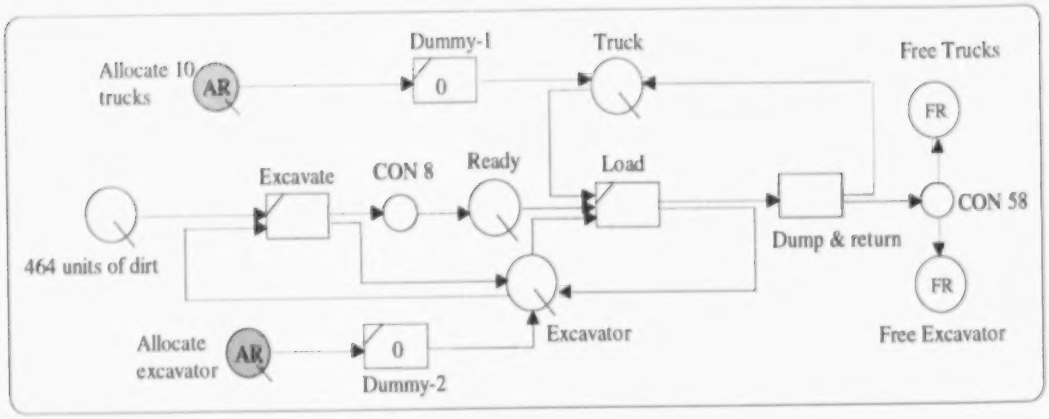


Figure 4-12: Process model for the excavation process

Figure 4-13 shows the process model for the piling process. This is a modular process that is referenced by Abutment-1, Abutment-2 and Pier-1.

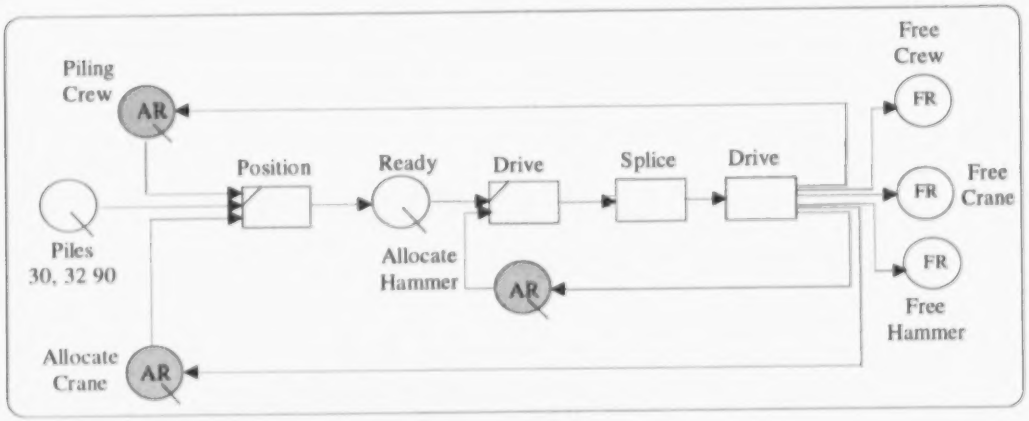


Figure 4-13: Process model for the piling process

In this process a crane, a diesel hammer and a piling crew are required. These resources are allocated to the piling process using the Allocate Resource Node and are freed after the completion of one cycle. The QUE node "Pile" shown in the figure is updated for the three different locations where the piling process is performed. Therefore, the modeler first references the piling process and then depending upon the location

(Abutment-1, Abutment-2 and Pier-1) initializes the appropriate number of sheet piles (30 piles for Abutment-1, 32 piles for Abutment-2 and 90 piles for Pier-1).

Figure 4-14 shows the process model for the cofferdam process. This process is also a modular process and is used by Pier 2, 3, 4 and 5. The work tasks in this process require a steel frame, a crane and a diesel hammer along with a piling crew. These resources are allocated by using three Allocate Resource Nodes.

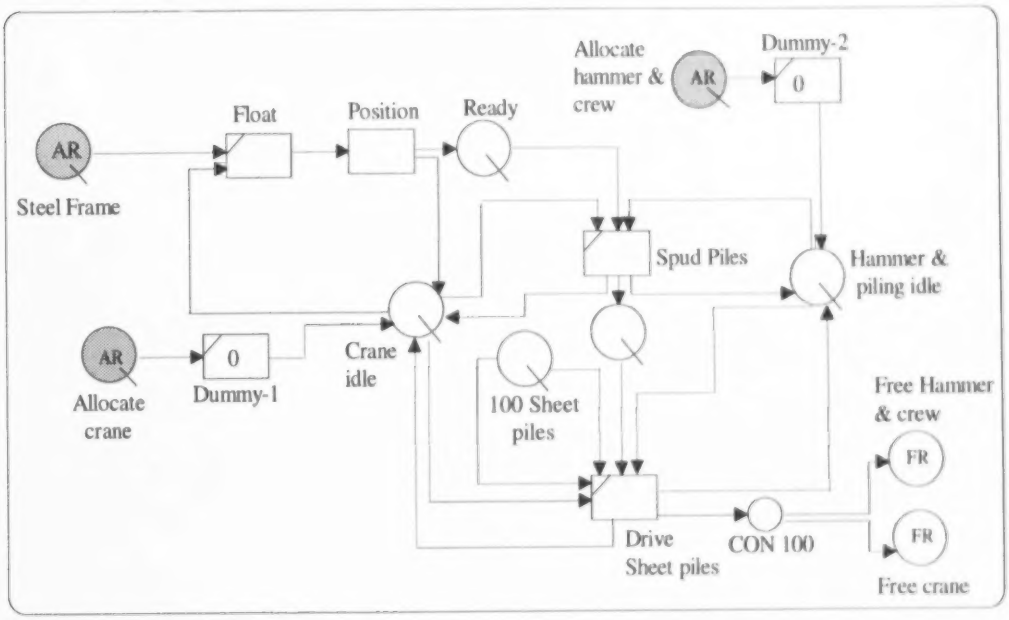


Figure 4-14: Process model for the cofferdam process

The steel frame for the cofferdam is allocated to the process at the first Allocate Resource Node. As the cofferdam process is performed in the river, the resources required for the process are freed only after the process is completed. This scenario is modeled by placing two dummy COMBI nodes after the Allocate Resource Node for the crane and diesel hammer. These resources are released after the completion of the process at the Free Resource Node that are placed after the CONSOLIDATE function.

Figure 4-15 shows the process model for the "blinding layer" process. In this process three resources are required. The concrete is delivered to the site by transit mixer trucks and is then placed using a concrete pump and concreting crew. The concrete crew and concrete pump resources are again modeled as "slave" resources and are released only after the completion of the specific concreting process at each pier.

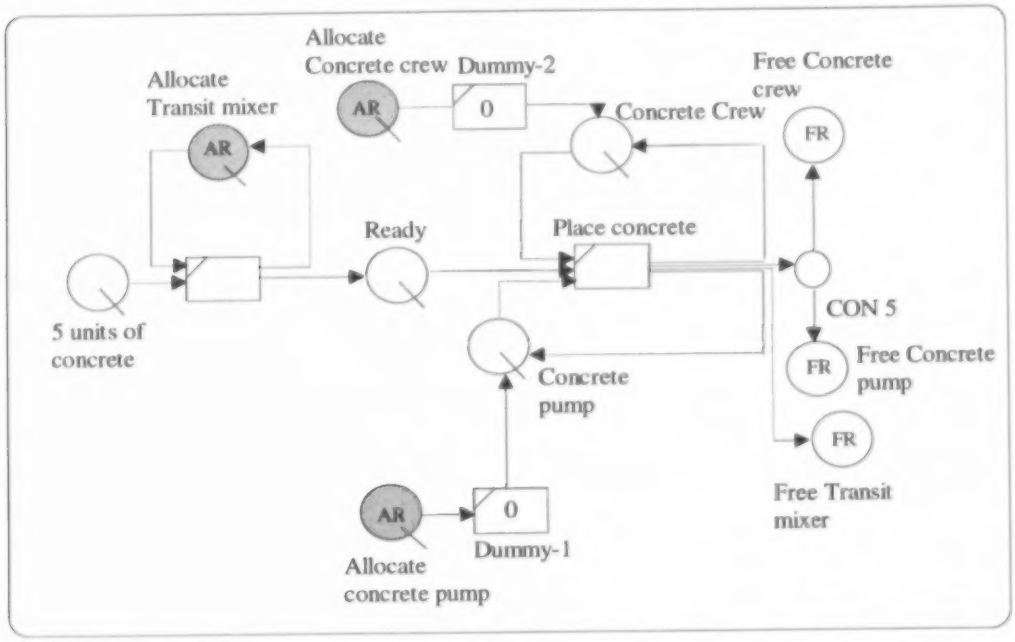


Figure 4-15: Process model for the blinding layer process

Figure 4-16 shows the process model for the footing formwork process. This process is a simple process in which the details of various work tasks are not modeled.

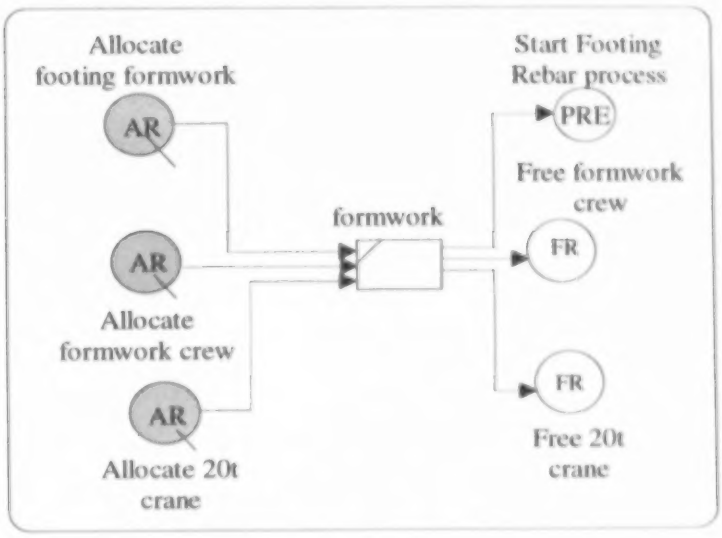


Figure 4-16: Process model for the formwork process

This process model was developed based on the assumption that the modeler was not directly responsible for the details of the formwork process. In this process model the three resources namely: one set of formwork, one formwork crew and one crane, are each allocated to the process for two days. These resources are assumed to be freed only after the completion of the formwork process. Such a process model allows the modeler to focus on the processes that are important without neglecting the effect of the processes that are not as important. Figure 4-17 shows the process model for the footing rebar process. This is also a simple process model and uses the same philosophy as the formwork process.

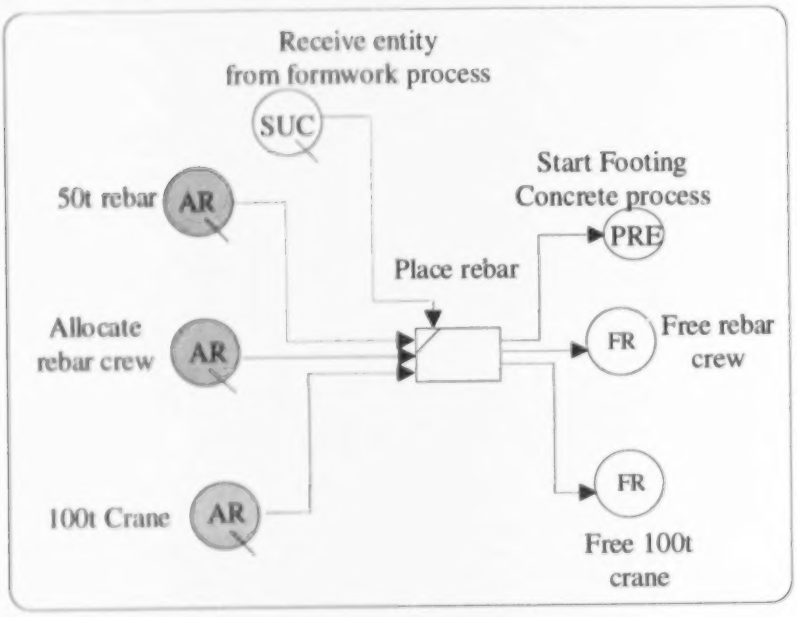


Figure 4-17: Process model for the rebar process

Figure 4-18 shows the process model for the concreting process for the pier footing. This process model can only be started after the completion of the rebar process. This situation is modeled by including a successor node that references the predecessor node in the rebar process. The remaining details of the process are similar to the concreting in the blinding layer.

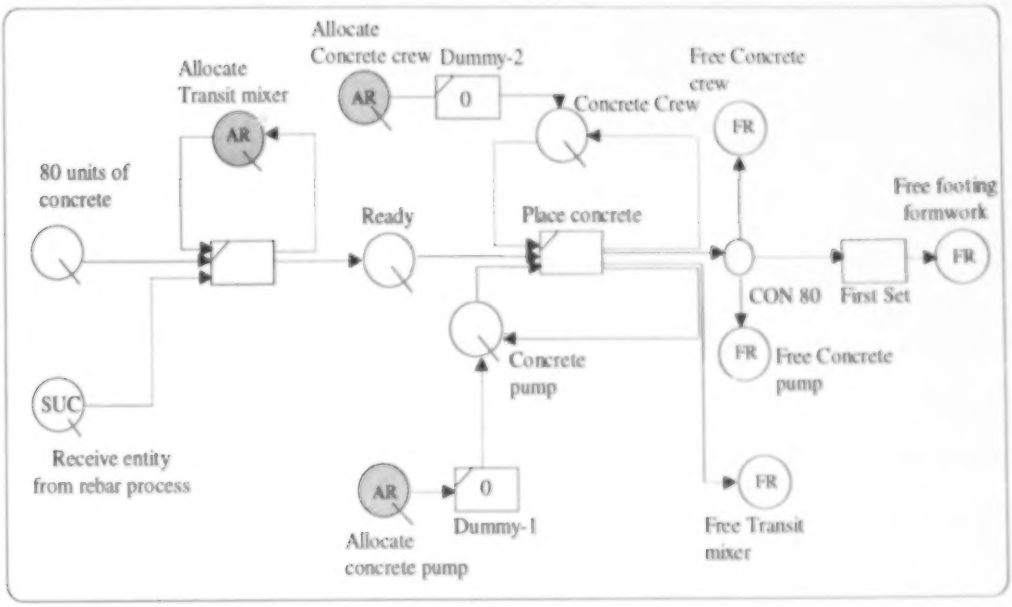


Figure 4-18: Process model for the concreting process

This section outlined the various process models that were used in the example project. The process models that are repeated and are similar to the ones explained here are not described. These models were duplicated in the background by the prototype when the simulation was performed.

4.6 SIMULATION OF THE BRIDGE PROJECT

The information provided in Section 4.3, 4.4 and 4.5 constitutes the project level simulation models for the sub-structure of the Peace River Bridge project. These models were translated using the HSM prototype system. In its current form the prototype does not provide a completely automated translation of the simulation models.

It is normal practice in simulation experimentation to use deterministic duration for the initial experiment. This is commonly called the "pilot run". The author performed the

pilot run for the bridge project using deterministic duration estimates for all the work tasks. A project completion time of 36.91 weeks based on a seven work day week and one eight hour shift per day was obtained from this simulation run. The detailed results of the pilot run are tabulated in Table C-2 in Appendix C. In this table the processes are grouped by abutments and piers and the process start and finish times and durations are provided. In order to validate the results obtained from the pilot run a CPM schedule for the project was developed using Primavera Project Planner (Primavera, 1990). In this schedule each process was treated as an activity and the construction logic similar to the one defined by the operation sequencing was used. The project completion obtained from this analysis was 36 weeks. The project completion time from the pilot run was approximately the same as the project completion time obtained from the CPM analysis. This validated the simulation models for the Peace River bridge project developed by the author. In the simplest form HSM provides results that can be normally obtained by CPM or PERT. However, HSM provides features that not only makes planning realistic but also allows the modeler to perform simple tasks like optimal resource allocations, selection of an implementation strategy and selection of construction methods.

After the validation of the deterministic case the simulation models were embellished to include stochastic duration estimates. For the purpose of this experiment triangular duration estimates were used and thirty simulation runs were performed. This experiment resulted in a mean project completion time of 40.6 weeks with a standard deviation of 0.949. Table C-3 in Appendix C provides the summary of process times for this

experiment. This experiment is termed as the "base case" experiment and the experimentation explained in the following sections were based on this model.

4.7 EFFECT OF CHANGING THE RESOURCE ALLOCATIONS

In order to demonstrate some of the potential benefits of HSM a sensitivity study was performed by varying the quantity of a selected resource. In this experiment the resource "CRANE 100t WEST" was selected. To study the effect of changing the resource allocations the following scenarios were developed:

1. Scenario-1: one "CRANE 100t WEST" was allocated to the project.
2. Scenario-2: two "CRANE 100t WEST" were allocated to the project.
3. Scenario-3: three "CRANE 100t WEST" were allocated to the project.

Using these allocations the simulation experiment was repeated while the other information was the same as for the base case. It was assumed that the objective of this study was to determine the effect of these allocations on the project completion time. The experiment did not include the study of the associated cost of these allocations. The summary of the results of this study is provided in Table 4-4.

Table 4-4: Summary of the results for different resource allocations

Scenario	Resource Capacity	Project Duration	95% confidence interval
1	1	42.1 weeks	(41.98, 42.21)
2	2	40.6 weeks	(40.26, 40.94)
3	3	40.5 weeks	(40.18, 40.82)

The above results were obtained by simulating each scenario for 30 independent runs. Using the sample standard deviation a 95% confidence interval was calculated for the

three scenarios. On the basis of the confidence intervals it was inferred that the experimentation was sensitive to the change in the number of resources allocated to the project.

It can be seen from the results that by using two cranes on the project instead of one the project completion time decreases by 3.6 %. Considering the cost of an extra crane and the resulting savings the modeler can select an optimal resource allocation for the project. The increase in capacity of the resource to 3 changes the project completion time to 40.5 weeks. It can be observed that use of three cranes on the project does not produce significant reduction in the project completion time. Based on the project objectives such an experimentation can provide the modeler with useful insights for the allocation of resources and their effect on the project. The detailed results of this experiment are attached in Appendix C.

4.8 EFFECT OF CHANGING THE OPERATION SEQUENCING

On a construction project the implementation strategy adopted can result in different project completion times and total cost. HSM provides the modeler with features that allow experimentation with different implementation strategy. Such an experimentation is performed by simply revising the operation sequencing. To study the effect of operation sequencing the following scenarios were used:

1. Scenario-4: this scenario is the same as the base case scenario.
2. Scenario-5: this scenario is similar to the base case scenario except that it has a revised operation sequencing as shown in Figure 4-19.

As shown in Figure 4-19 the revised sequencing allowed the construction to start on the two abutments and six piers simultaneously. In essence all the processes compete for the common resources. Such an experimentation is easily possible in HSM because it is driven by simulation. Traditional tools like CPM would require substantial work for such a comparison and would actually involve development of a new CPM diagram for the revised implementation strategy. Table 4-5 summarizes the results from the two scenarios.

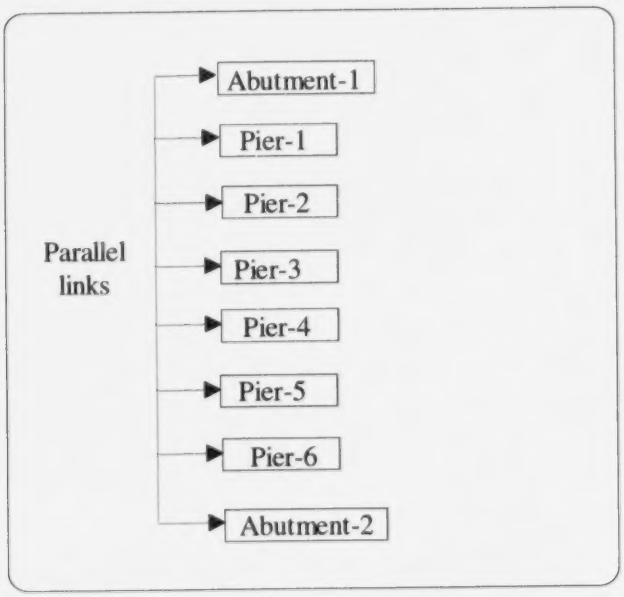


Figure 4-19: Revised operation sequencing

Table 4-5: Summary of results for revised operation sequencing
(a) Project completion time statistics

Project Completion Time	Mean Value	95% Confidence Interval	Minimum Value	Maximum Value
Scenario-4	40.6 weeks	(40.26, 40.94)	38.8 weeks	42.4 weeks
Scenario-5	38.8 weeks	(38.47, 39.13)	37.3 weeks	40.7 weeks

(b)Utilization of key resources

Resource	Average utilization for scenario-4	Average utilization for scenario-5
Diesel Hammer	0.61	0.64
Pile Crew	0.61	0.64
Truck West	0.22	0.23
Formwork Crew West	0.26	0.28
Rebar Crew East	0.43	0.45
Pedestal Form East	0.30	0.31

It can be seen from the summary of results that by revising the operation sequencing the mean project completion time is reduced to 38.8 weeks. This reduction in project completion time is achieved by an increase in the average utilization of the key resources as shown in Table 4-5(b). For example the average utilization of the diesel hammer increased from 61 % to 64 % (3 % increase). The average utilization shown in the table is a theoretical value that is based on the total project completion time.

4.9 SIGNIFICANT RESULTS

From the experimentation conducted for the Peace River bridge project, it was found that:

1. The graphical user interface of HSM was satisfactory and greatly improved the solicitation of the project information from the user.

2. The modeling framework of HSM was found to be adequate for modeling a real construction project. The study demonstrated that project level simulation is feasible.
3. The experimentation discussed in this chapter demonstrated some potential benefits of a simulation-based project planning method.

4.10 CONCLUDING REMARKS

This chapter described the case study used in this research was provided. Various experiments performed using HSM were explained. An explanation of the key modeling concepts of HSM, a validation of these modeling concepts and some potential benefits of using a simulation based system was provided.

Chapter 5: Final Discussion

5.1 SUMMARY OF THE RESEARCH

Traditional planning tools like CPM and PERT induce inherent deficiencies in the project plan due to their underlying assumptions and limitations. They are founded on the premise that planning a project requires identifying the tasks, assigning deterministic durations to these tasks, linking them in a predecessor-successor relationship and superficially superimposing resource allocation. On the contrary a construction project is characterized by the random nature of the conditions under which it is implemented and by the dynamic nature of the resource utilization. Therefore, to realistically plan construction projects and produce accurate schedules it is essential to model these important aspects. This research has culminated into the development of a simulation-based method for planning of construction projects that incorporates the above mentioned aspects into the project plan. The method developed by the author is called Hierarchical Simulation Modeling (HSM) method. It utilizes hierarchical and modular constructs in the development of project level simulation models.

Under HSM methodology, a project is first divided into manageable components by identifying the operations, processes and work tasks involved in it. All the resources are then defined in a common resource library. The operations are linked to each other to define project implementation strategy and process models are developed using CYCLONE methodology. These steps have been automated in the prototype program

developed for HSM. HSM provides a rich graphical user interface that makes model definition for the project simple and efficient.

The method developed in this research is generic in nature and could be applied to the simulation of and large and complex system. The method lends itself to different types of "what-if" experiments the modeler wishes to try.

5.2 RESEARCH CONTRIBUTIONS

This research has resulted in the following major contributions:

1. Development of a framework for project level information representation that will facilitate computer simulation analysis.
2. Simplification of simulation modeling through graphical user interface.
3. Development of a hybrid programming method that will facilitate programming of any proposed simulation modeling framework.
4. Demonstration of the feasibility of project level simulation using HSM.

The method developed in this research has utilized the popular concepts of WBS and process modeling to arrive at a simple and effective technique. The evolution of HSM has resulted in enhancements to both these techniques.

Enhancements to the work breakdown structure has been achieved to better model the project scope. Through the use of modularity this enhancement has greatly simplified the task of planning.

Through enhancement to CYCLONE process modeling method an efficient and simple environment for simulating all the processes on a project has been achieved. The

CYCLONE method has been enhanced to provide features like multiple process interaction and resource identification.

The research conducted in this work has also lead to the development of an effective and structured method of modeling project information. The implementation of the method has provided insight into the development of a simulation environment using object oriented concepts and event driven programming.

5.3 CONCLUDING REMARKS

HSM provides an efficient and realistic approach to planning of construction projects. The need of such a modeling method was demonstrated earlier by AbouRizk and Dozzi (1993). AbouRizk and Dozzi stated that "*for simulation modeling technique to be useful at the project level the hierarchical development should be implemented as part of the modeling methodology and the programming implementations.*"

HSM uses hierarchical concepts to structure the project information in an efficient manner. It allows the modeler to break individual operations to different levels of detail, hence providing increased modeling versatility. At the top level it provides a higher level of abstraction and a complete plan without a great deal of detail. At the lowest level it provides a high degree of modeling detail in the form of process models. This feature also allows different personnel to attain appropriate information from the same project plan.

Reusability and assembly of modular operations and processes is a feature that makes HSM attractive for construction practitioners. A construction company can develop modular models describing the general construction methods adopted and then reuse these models on similar projects. These generic modular models could also be assembled with

each other to produce more complex and detailed models based on the requirements of various projects.

HSM, unlike methods developed by Dabbas (1981) and Odeh et. al. (1993), does not borrow any concepts from traditional planning tools like CPM. It is a system that is driven by simulation. This allows it to inherit many of the benefits attainable through simulation, including dynamic portrayal of the project plan, utilization of stochastic durations, incorporation of external factors like weather and equipment breakdown. It also allows the modeler to perform sensitivity studies involving resource usage. In HSM the modeler depicts the implementation strategy for the project in the form of operation sequencing. The operation sequencing feature in HSM allows the modeler to experiment with various implementation strategies by altering the sequencing in a very simple fashion.

Another major enhancement afforded by HSM is the availability of cyclic links between one or more operations. This feature makes HSM versatile and applicable on a wide range of construction projects. For a linear project, like a pipeline project, the modeler can identify the basic operation required in performing the work for a unit length and then cycle these operations using the cyclic link.

As illustrated in the discussion on the example project, HSM allows the modeler to experiment with the project plan more readily as compared to the existing tools. To model external factors like weather, learning effect or effects of management decisions, the modeler would be required to run different scenarios and come up with the optimum solution.

Through the development of HSM the author has demonstrated that a realistic solution to the project planning problem may be attained by the use of simulation and that the current process modeling orientation of construction simulation needs to be enhanced to perform this task. The author envisions that the proposed method will prove to be an efficient tool for project planning. The key factors for this efficiency are as follows:

1. the method allows demarcation of the project information into logical and manageable packages
2. the method is driven by the dynamic allocation and utilization of resources
3. the method allows advanced facilities not afforded by traditional tools like CPM and PERT. For example HSM unlike the traditional tools allows cycling of operations and reusable modular model components.

5.4 RECOMMENDATIONS

The development of the HSM modeling concepts and HSM prototype was directed towards demonstration of the feasibility of a simulation based planning tool for construction. In the current form HSM has the following limitations:

1. HSM currently does not utilize calendar day scheduling. In order to overcome this limitation calendar day scheduling algorithms available in literature can be utilized.
2. The concept of work shifts is not included in the current version of HSM. This limitation can be overcome by the use of a GATE (SLAMSYSTEM, 1990) node.

3. Currently HSM assumes that all the resources that are initialized in the resource library at the start of the project are retained at the site till the completion of the project. It is therefore essential to include a release mechanism that will allow the modeler to release a resource from the project before the project completion.

4. The major limitation of HSM is that it does not include an internal simulation module. Use of SLAMSYSTEM as the simulation engine reduces the seamless operation of the prototype tool. In order to achieve efficient results it is essential to develop an internal simulation module that is geared towards the HSM modeling concepts.

5. HSM prototype is not "intelligent" to trap modeling errors that are induced by the modeler in a project plan. The object oriented concepts described in this work can be enhanced to produce a tool that can guide the modeler in the plan development. In the future it would be possible to include "intelligent" features in the internal structure of the modeling elements. For example, the process modeling elements in HSM could have a feature that detects the type of follower defined for that element. Using this feature it would be very simple to check the precedence rules defined for process modeling.

6. The reporting module of HSM does not produce reports that are commonly utilized in the industry. Currently the modeler is required to infer this information from the output produced by SLAMSYSTEM. An advanced reporting module compatible with the internal simulation engine of HSM will eliminate this limitation.

Bibliography

AbouRizk S. M. and Dozzi S.P. (1993). "Applications of Simulation in Resolving Construction Disputes" Journal of Construction Engineering and Management, ASCE, 119(2), 355-373.

AbouRizk, S. M. (1993). Technical discussion on: "Modeling Operational Activities in Object-oriented Simulation", by A. A. Oloufa. Journal of Computing in Civil Engineering, ASCE, in press.

AbouRizk S. and Sawhney A. (1994) "An information modeling methodology for simulation based project planning", Annual Conference of the Canadian Society of Civil Engineering, C.S.C.E., Winnipeg, Manitoba, June 1994, (in press).

Ahuja, N.H., and Nandakumar, V., (1985). "Simulation model to forecast project completion time." Journal of Construction Engineering and Management, ASCE, 111(4), 325-342.

Ang, A.H.S., Abdelnom, J. and Chaker, A.A. (1975) "Analysis of Activity Networks under Uncertainty", Journal of Engineering Mechanics Division, ASCE Vol. 101, No. EM4, pp 373-387.

ATU (1989). "Alberta Transportation and Utilities (Bridge Engineering Branch) - Peace River bridge plans and specifications", Edmonton, Alberta.

Carr R.I. (1979), "Simulation of construction project duration", Journal of Construction Engineering and Management, American Society of Civil Engineers, Vol. 105 No. CO2, pp 117 - 128

Chang, D. Y. and Carr, R. I. (1987). "RESQUE: A Resource Oriented Simulation System for Multiple Resource Constrained Processes" Proceedings of the PMI Seminar/Symposium, Milwaukee, Wisconsin, 4-19.

Cox B.J. (1987) "Object-oriented Programming - An Evolutionary Approach", Second Edition, Addison-Wesley Publishing Company, MA.

Crandall K.C. (1976), "Probabilistic time scheduling", Journal of Construction Division, American Society of Civil Engineers, Vol. 102 No. CO3, pp 415- 423.

Crandall, K.C., (1977). "Analysis of schedule simulations." Journal of Construction Engineering and Management, ASCE, 103(C03), 387-394.

Dabbas M. (1981) "Computerized Decision making in Construction", Ph.D. thesis presented to the Georgia Institute of Technology, Atlanta, Georgia, in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Dabbas M. and Halpin D. W. (1982). "Integrated Project and Process Management" Journal of the Construction Division, ASCE Vol. 109 No. CO1:361-373.

Douglass, D.E., (1978). "PERT and simulation." Winter Simulation Conference proceedings, IEEE, vol. 1, pp 88-98.

Eardley, V.T., and Murphee, Jr., E.L. (1969) "Production of multi-level critical path networks." Construction series, No.13. Urbana: Department of Civil Engineering, University of Illinois, August, 1969.

Elmore R.L. and Sullivan D.C. (1976) "Project control through work packaging concepts", Transactions of the American Association of Cost Engineers, 20th Annual Meeting, Boston, Massachusetts, July 18-21, 1976, pp 50-56.

Halpin D., AbouRizk S., and Hijazi A. (1989). Sensitivity Analysis of Construction Operations. Proceedings of the 7th National Conference on Microcomputers in Civil Engineering, Orlando Florida. 181-185

Halpin D.W., Escalano A.L. and Szmurlo P.M. (1987) "Work packaging for project control", Source Document, The Construction Industry Institute, The University of Texas at Austin, Austin, Texas.

Halpin, D. W. (1973). "An Investigation of the Use of Simulation Networks for Modeling Construction Operations" Ph.D. thesis presented to the University of Illinois, at Urbana-Champaign, Illinois, in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Halpin, D. W. (1976). "CONSTRUCTO-An Interactive Gaming Environment" Journal of the Construction Division, ASCE, 102 No. CO1:145-156.

Halpin, D. W. (1977). "CYCLONE: Method for Modeling of Job Site Processes" Journal of the Construction Division, ASCE, 103(3):489-499

Halpin, D. W. (1990). "MicroCYCLONE User's Manual" Division of Construction Engineering and Management, Purdue University. West Lafayette, Indiana.

Halpin, D. W. and Riggs L.S. (1992) "Planning and Analysis of Construction Operations" Wiley Interscience. New York, N.Y.

Halpin, D. W. and Woodhead, R (1976) "Planning and Analysis of Construction Operations" Wiley Interscience. New York, N.Y.

Heathington, K.W., and Chatterley, J.L. (1971) "Computer simulation in design and construction." Professional Engineer, April 1971.

Johnston, R.E. (1980) "Discrete Event Simulation as a Management Tool for Planning and Scheduling of Overhauls of U.S. Navy Ships", unpublished master's project, Purdue University.

Kim, T.G. and Zeigler B.P. (1987) "The DEVS Formalism: Hierarchical, Modular Systems Specification in an Object-Oriented Framework. In Proceedings of 1987 Winter Simulation Conference.

Knapp K.E. (1987) "The Smalltalk Simulation Environment, Part II", Proceedings of the 1987 Winter Simulation Conference, 146- 151.

Kostetsky, O. (1986). "A simulation approach for managing engineering projects." Proceedings 1986 International Conference on Robotics and Automation, IEEE, 318-324.

Law, A.M. and Kelton W.D. (1991) "Simulation Modeling and Analysis", McGraw-Hill, NewYork.

Lee, S.M., Moeller G.L. and Digman, L.A., (1982) "Network Analysis for Management Decisions: A Stochastic Approach", Kluwer- Nijhoff Publishing, Boston, MA.

Liang,Y. Liu, and Ioannou, P.G. (1992) "Graphical object-oriented discrete-event simulation system." Proceedings of the 1992 Winter Simulation Conference 1285-1291.

Liu L.Y., and Ioannou P.G., (1992). "Graphical Object-Oriented Simulation System for Construction Process Modeling" Proceedings of the Eighth Conference on Computing in Civil Engineering, ASCE, Dallas, Texas. 1139-1146.

Lluch J. F., and Halpin D.W. (1981). Analysis of Construction Operations Using Microcomputers. Journal of the Construction Division, ASCE Vol. 108 No. CO1:129-145.

Luna, J. (1990). "Object-Oriented Multi-Simulation Environment." In Proceedings of the 1990 Winter Simulation Conference.

Luna, J. (1991). Object Framework for Application of Multiple Analysis Methods. Object-Oriented Simulation 1991, Simulation Series Volume 23, Number 3, 81-86.

Luna, J. (1992) "Hierarchical, modular concepts applied to an object-oriented simulation model development environment." Proceedings of the 1992 Winter Simulation Conference 694-699.

Lutz J.D., (1990). Planning of Linear Construction Using Simulation and Line of Balance. Ph.D. Dissertation, School of Civil Engineering, Purdue University, W. Lafayette, IN.

MacCrimmon, K.R. and Ryavec, C.A. (1964) "An Analytical Study of the PERT Assumptions", Operations Research, Vol. 12, pp 16-38.

McKenney, J.L. (1967) "A clinical study of the use of simulation model." The Journal of Industrial Engineering. Vol.XVIII, No 1, Jan 1967.

Microsoft (1993c). *Microsoft Access - Programmer's Guide. Version 2.0*, Microsoft Corporation, Redmond, WA.

Microsoft (1993a). *Microsoft Visual Basic - Programmer's Guide. Version 3.0*, Microsoft Corporation, Redmond, WA.

Microsoft (1993b). *Microsoft Visual C++ - User's Guide. Version 1.0*, Microsoft Corporation, Redmond, WA.

Moder, J., Phillips, C.R. and Davis E.W. (1970) "Project Management with CPM, PERT and Precedence Diagramming, Third edition, Van Nostrand Reinhold, New York.

Murphy R.H., (1981) "Developing work packages for project control", Proceedings of Specialty Conference on Effective Management of Engineering Design, ASCE, Engineering Management Division, ASCE publications, New York, N.Y.

Odeh A.M., Tommelein I.D., and Carr R. I. (1992). "Knowledge-Based Simulation of Construction Plans" Proceedings of the Eighth Conference on Computing in Civil Engineering, ASCE, Dallas, Texas. 1042-1049.

Oloufa A.A. (1993) "Modeling Operational Activities in Object-oriented Simulation", Journal of Computing in Civil Engineering., ASCE, Vol. 7 No 1, pp 94-106.

Papageorgiou, J.C. (1985) "An Application of GERT to Air Force Development Planning" Project Management: Methods and Studies, ed. by

Paulson, Boyd C., Jr. (1987). Interactive Graphics for Simulating Construction Operations. Journal of the Construction Division, ASCE, 104(1):69-76.

Pidd, M. (1992) "Object oriented & three phase simulation." Proceedings of the 1992 Winter Simulation Conference 689-693.

Primavera (1990). Primavera Project Planner - Reference Version 5.0, Primavera Systems Inc., BalaCynwyd, PA.

Pritsker A.A.B., Sigal, C.E. and Hammesfahr R.D.J. (1989) " SLAM II - Network Models for Decision Support", Prentice-Hall, Inc., NJ.

Pritsker, A. A. B. (1985), Introduction to Simulation and SLAM-II. John Wiley and Sons, Inc., New York, N.Y.

Pritsker, A.A.B. (1979), "Modeling and Analysis using Q-GERT Networks" Second edition, Wiley and Pritsker Associates, New York and West Lafayette, IN.

Pritsker, A.A.B. (1986), "Introduction to simulation and SLAM II" Second edition, Wiley and Pritsker Associates, New York and West Lafayette, IN.

Riggs, L.S. (1980). Simulation of Construction Operations. Journal of the Construction Division, ASCE Vol. 106 No. CO1:145-163.

Riggs, L.S. (1989). "Risk Management in CPM Network", Microcomputers in Civil Engineering, Elsevier Applied Science, Vol. 3, No 3 pp 229-235.

Roberts, S.D. and Heim J. (1988) "A Perspective on Object-oriented Simulation", Proceedings of the 1988 Winter Simulation Conference 277-281.

Rothenberg J. (1986) "Object-oriented Simulation: Where do we go from here?", Proceedings of the 1986 Winter Simulation Conference.

Sanderson D.P., Sharma R., Rozin R. and S. Treu, (1991), "The hierarchical Simulation Language HSL: A versatile Tool for Process-Oriented Simulation", ACM Transactions on Modeling and Computer Simulation, 113-153.

Teicholz, P. (1963). A Simulation Approach to the selection of Construction Equipment. Technical Report No. 26, The Construction Institute, Stanford University.

Touran, A. (1981). Construction Operations Data Acquisition and Processing Via Time-Lapse Photography Interfaced to a Microcomputer. Ph.D. Dissertation, School of Civil Engineering, Stanford University, Stanford Ca.

Ulgen, O.M., Thomasma, and Y. Mao, (1989), "Object-Oriented Toolkits for Simulation Program Generators.," Proceedings of the 1989 Winter Simulation Conference.

van Slyke, R.M. (1963) "Monte Carlo Methods and the PERT problem", Operations Research, Vol. 11, pp. 839-860.

Woolwry J.C. and Crandall K.C. (1983), "Stochastic network model for planning scheduling", Journal of Construction Engineering and Management, American Society of Civil Engineers, Vol. 109, No. 3 pp 342 - 354.

Wortman, D.B. and Sigal C.E. (1978) "Project Planning and Control using GERT", Pritsker and Associates, West Lafayette, IN.

Yancey, D.P. and Musselman K.J. (1980), "Critical Statistics in General Project Planning Networks", Pritsker and Associates, West Lafayette, IN.

Zeigler, B.P.(1984) "Multifacetted Modeling and Discrete Event Simulation." Academic Press Inc., Orlando, Fl.

Zeigler, B.P.(1986) "Hierarchical Modular Modeling/Knowledge Representation", Proceedings of the 1986 Winter Simulation Conference, 129-137.

Zeigler, B.P. (1987). "Hierarchical, Modular Discrete-Event Modeling in an Object-Oriented Environment", Simulation, November 1987, 219-230.

Appendix A: Implementation details of HSM prototype

INTRODUCTION

HSM prototype provides a graphical user interface for solicitation of project information from the user. After the project information has been provided HSM translates this information into a project level simulation model which is then simulated using SLAMSYSTEM. The HSM modeling framework was described in Chapter 2 and the description of the HSM prototype was provided in Chapter 3. In this appendix details of the computer implementation of HSM prototype are provided. In the first section a description of the development of the modeling elements as C++ objects is provided. The second section is devoted to the description of the use of these modeling elements in Visual Basic for the development of the graphical user interface. Third section provides the data structure utilized in HSM prototype for storage of the project information.

C++ IMPLEMENTATION OF HSM MODELING ELEMENTS

HSM prototype utilizes eleven modeling elements that are shown in Table A-1. The table lists the modeling elements along with their graphical representation and name of the "custom control" files. Custom controls are special types of C++ classes that can be used to extend the Visual Basic programming environment. A custom control file is similar to a Windows dynamic link library that is designed to interact with Visual Basic. Though it is possible to include more than one custom control in a *.vbx file ("*" is a wildcard character that is a place holder for a user given name), the author has developed individual VBX files for each modeling element for simplicity. These *.vbx files are required during design of the program as well as at run time.

Table A-1: Custom control files for HSM modeling elements

Element	*Graphical format*	*Custom Control File*
OPERATION		opr.vbx
PROCESS		mypro.vbx
ALLOCATE RESOURCE		arn.vbx
FREE RESOURCE		frn.vbx
PREDECESSOR		pred.vbx
SUCCESSOR		sucd.vbx
NORMAL		norm.vbx
COMBI		comb.vbx
QUE		que.vbx
FUNCTION		consd.vbx
COUNTER		count.vbx

The program code to be written for a custom control takes care of the following issues:

1. appearance of the control on the screen.
2. properties of the control including graphical properties or any special properties.
3. events and methods associated with the custom control.
4. behavior of the control within the Visual Basic environment both during design time and run time.

For HSM the eleven modeling objects described in Table A-1 were programmed in Visual C++. The program code for each modeling element has been broken down into separate files. These files are then compiled and linked together to develop the *.vbx files for each control. Table A-2 lists the files that are associated with each control and provides a description of each file.

Table A-2: Custom control source code files

Name of the file	Description
<name>.c	This file contains all control procedures for initialization and messaging of the control.
<name>.h	This file contains model, property and event declarations for the control.
<name>.def	This is a module definition file for the custom control.
<name>.rc	This file identifies the bitmap files used for the control and assigns ID numbers.
<name>cu.bmp	Icon to display the control in an unselected position for VGA monitors.
<name>cd.bmp	Icon to display the control when it is selected by the user for VGA monitors.
<name>mu.bmp	Icon to display the control in an unselected position for monochrome monitors.
<name>eu.bmp	Icon to display the control in an unselected position for EGA monitors.
makefile	Required by the C++ environment for compilation and linking of the above files.

PROGRAMMING THE GRAPHICAL USER INTERFACE

The graphical user interface of HSM prototype has been developed in Visual Basic. In the development process Visual Basic was extended by importing the modeling elements developed in Visual C++.

Visual Basic is a programming environment for the Microsoft Windows and OS/2 Presentation Manager systems. The Visual Basic language uses the simplified syntax of BASICA and GW-BASIC, and supports nearly all of their capabilities. Programming in Visual Basic is centered around the objects called VB Objects which include:

Forms: These are windows that act as templates for the entire program as well as other VB objects. Forms have a set of pre defined properties, events and methods. A program developer can use these to customize the graphical user interface.

Figure A-1 illustrates a typical form object in Visual Basic program.

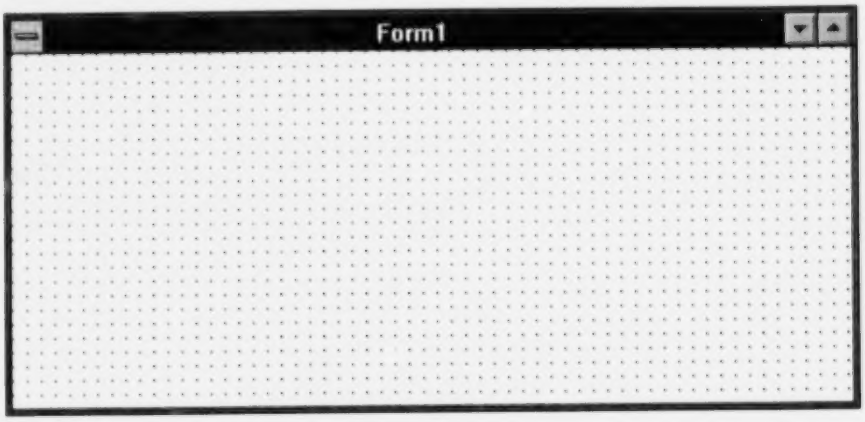


Figure A-1: Visual Basic Form Object

Control Objects: These are the graphical objects which can be drawn on a form object to produce the graphical user interface.

Figure A-2: Control Objects in Visual Basic

Figure A-2 illustrates the various control objects that are available to a program developer in the Professional edition of Visual Basic. For each type of object, Visual Basic pre-defines a set of events that can be used to respond to by writing code. It is easy to respond to events since form objects and control objects have the built-in ability to recognize user actions and invoke the code associated with them.

Figure A-3 shows the pre-defined events that the form object can recognize.

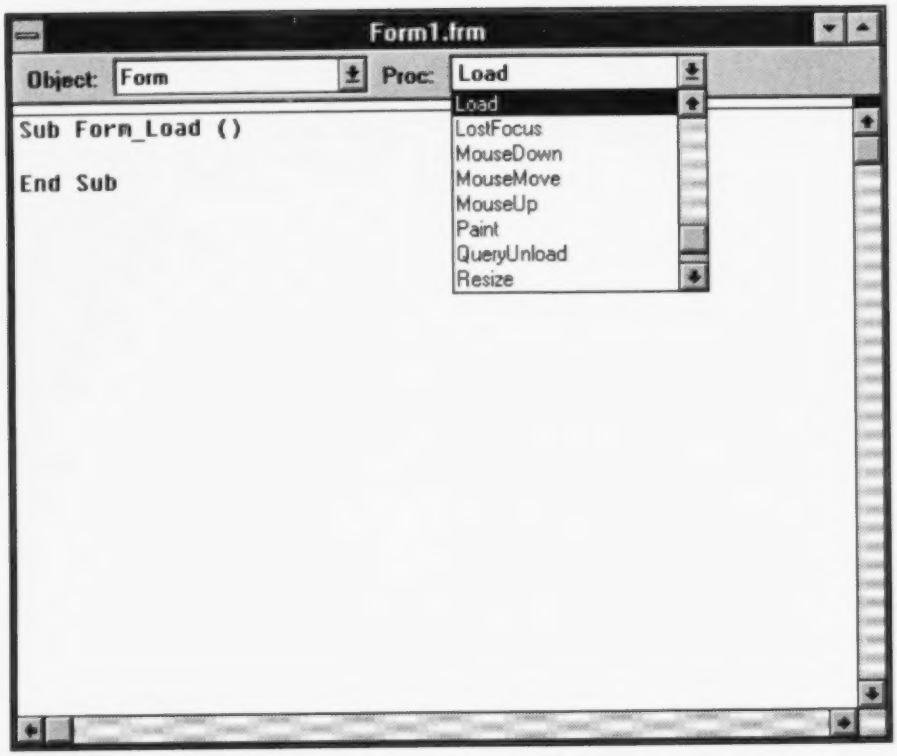


Figure A-3: Events associated with the Form Object

Similarly a control object in Visual Basic e.g. a textbox has a set of pre-defined properties which the program developer can use to create a graphical user interface. Figure A-4 illustrates some of these pre-defined properties.

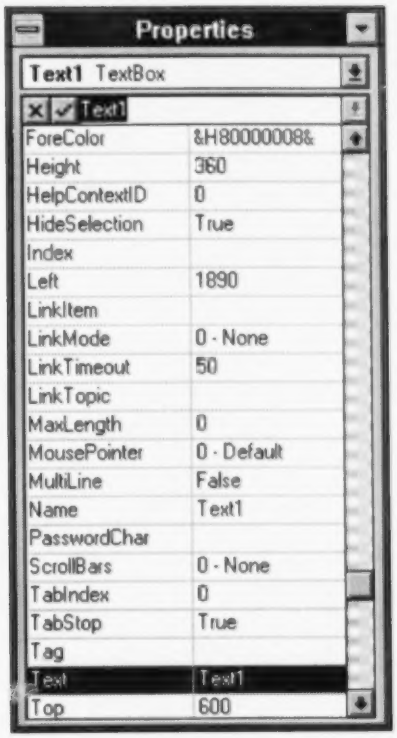


Figure A-4: Properties associated with a Control Object

In the development of HSM the above shown control objects were extensively utilized. But the key feature of the implementation was the use of imported custom controls specifically designed for the graphical user interface of HSM. Figure A-5 shows the tool bar of Visual Basic after the HSM modeling elements were imported into Visual Basic.

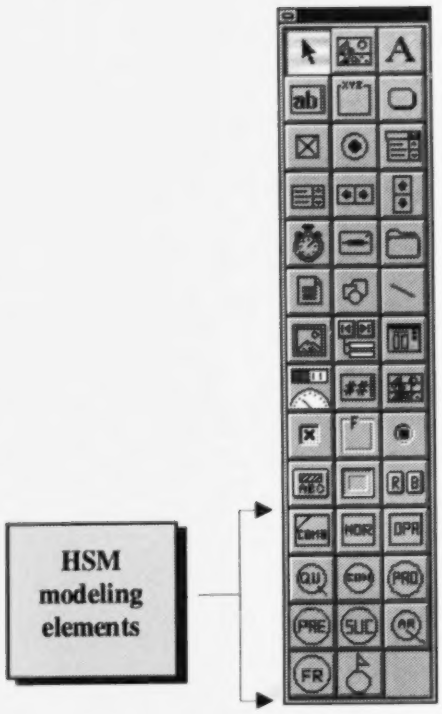


Figure A-5: Visual Basic toolbar with HSM modeling elements

The graphical user interface of HSM can be divided into four main sessions that are as follows:

1. Work breakdown structuring.
2. Resource management.
3. Operation sequencing.
4. Process modeling.

The structure of the Visual Basic program developed is as such centered around these sessions. The design of the program is such that there is a main template that is associated for each of the sessions. The user performs all the functions related to the particular session upon this template. The templates are supported by forms that are basically input

forms. This programming style has modularized the HSM prototype such that the user finds a natural flow of the various functions that are required to be performed during an HSM session.

To illustrate this style of programming, the programming required for the first session is used. A Visual Basic form named "ProjForm" acts as the template for the work breakdown structuring. This template provides the user with a pull down menu and drawing area upon which the operations and processes for the project can be defined. The Visual Basic code pertaining to this part of the HSM program is as such attached to this form. The "ProjForm" which is a template for the work breakdown structuring is supported by other forms and procedures. For example to solicit the information regarding an operation the "OprEdit" form is used. Subroutines specifically designed for the operation definition are attached to the "OprEdit" form and are used to work in conjunction with the "ProjForm". Similarly the form "ProcessForm" and its Visual Basic code is utilized to handle the definition of processes in the project WBS. This part of the code is attached in the attachment titled "Visual Basic code for the work breakdown structuring" at the end of this appendix. The remaining sessions are also programmed using a similar philosophy.

DATA STRUCTURE FOR STORAGE OF PROJECT INFORMATION

HSM allows the user to store a project plan for future use. In order to allow this, it was essential to design a data storage scheme for the project information. HSM prototype currently stores all the project information in a database that has been developed using

Microsoft Access (Microsoft, 1993c). The information stored in the project database is utilized for the following two tasks:

- to perform the translation and simulation of the project.
- to allow the user to open an existing project plan for latter use.

A blank file containing the empty structure of the project database is stored in the HSM directory. When a new project is started a copy of the blank database is provided for the project. The project information provided by the user during a session of HSM is automatically stored in this database. The database contains fifteen tables that store information pertaining to different aspects of the project plan. Table A-3 lists these tables and provides a brief description of the information stored in each table. A detailed description of the individual tables is provided in attachment titled "Structure of the project database".

Apart from the project database, HSM utilizes two other databases. The first database is utilized to store the information pertaining to "modular operations". The second database is utilized to store the information pertaining to "modular processes". Both these databases are stored in separate files that are saved in the same directory as the HSM executable files. Since both these databases store information that is similar to the project information, the structure is as such similar. The modular operation database contains the tables that store information pertaining to the operations and process while the modular process database contains tables that store the information pertaining to the processes. Table A-4 lists the tables that exist in the two databases.

Table A-3: List of the tables in the project database

Name of the table	*Description*
AllocateTable	This table stores information pertaining to resource allocations made at the process level.
AmFmTable	This table stores information pertaining to resource utilization at the process level. It stores details of the "allocate" and "free" nodes.
ConnectionTable	This table stores the information pertaining to the connections between nodes in a various process models.
CrewTable	This table stores information pertaining to crew combinations.
FinalRelation	This table stores the information derived after the translation of the project. It contains operation and process relations at the lowest level.
FinishTable	This table stores information regarding the finish of each operation in the project WBS.
FollowTable	This table stores information regarding the predecessor and successor nodes in various process models.
NodeTable	This table stores information related to various nodes in process models.
OprTable	This table stores the information pertaining to the operations in the project WBS.
Position	This table stores information pertaining to the graphical location of the operations on the WBS screen.
Position1	This table stores information pertaining to the graphical location of the processes on the WBS screen.
ProTable	This table stores information pertaining to the process in the WBS.
RelationTable	This table stores information pertaining to the operation sequencing.
ResourceTable	This table stores the information pertaining to the resources initialized for the project.
StartTable	This table stores the information derived from translation of the project regarding the start of various operations and processes at the lowest level.

Table A-4: Modular operation and modular process database

Modular operation database	*Modular process database*
OprTable	ProTable
Position	NodeTable
Position1	ConnectionTable
ProTable	
NodeTable	
ConnectionTable	

Visual Basic code for the work breakdown structuring

Source code in the file projform.frm pertaining to the work breakdown structuring

```
Dim AddProcess%, AddPro%
Dim DeleteProcess As Integer
Dim DeleteOperation As Integer•
Dim xoff As Integer
Dim yoff As Integer
Dim oldxop(100)
Dim oldyop(100)
Dim oldxpr(100)
Dim oldypr(100)
Dim ModOprProcessIndex(50)

Sub ConnectOpr ()
'*******************************************
'Subroutine to connect operations
'*******************************************
If AddOpr% = 1 Then
   MyLine.AddNew
   ChildX = Opr1(NOpr%).Left
   ChildY = Opr1(NOpr%).Top
   MyLine("x1") = ChildX
   MyLine("y1") = ChildY
   MyLine("Child") = Opr1(NOpr%).OprId
   MyLine("ChildType") = 2
   ParentIndex = OprEdit.Combo1.ListIndex
   If ParentIndex = 0 Then
         ParentX = 330
         ParentY = 38
         MyLine("x2") = ParentX
         MyLine("y2") = ParentY
         MyLine("Parent") = 0
   Else
         MyOpr.Index = "OprIdIx"
         MyOpr.Seek "=", ParentOperation(OprEdit.Combo1.ListIndex)
         CodeOfParent = MyOpr("Code")
         MyLine("x2") = MyOpr("OX")
         MyLine("y2") = MyOpr("OY")
         MyLine("Parent") = ParentOperation(OprEdit!Combo1.ListIndex)
   End If
End If
MyLine.Update
Call drawlines
End Sub

Sub ConnectProcess ()
```

```
'*****************************************
'Subroutine to connect Processes
'*****************************************
   MyLine1.AddNew
   ChildX = Process1(NProcess%).Left
   ChildY = Process1(NProcess%).Top
   MyLine1("x1") = ChildX•
   MyLine1("y1") = ChildY
   MyLine1("Child") = Process1(NProcess%).ProcessId
   ParentIndex = ProcessForm.Combo1.ListIndex
   MyTemp = ProcessParent(ParentIndex)
   If ProcessForm.Combo1.ListIndex = 0 Then
         ParentX = 330
         ParentY = 38
         MyLine1("x2") = ParentX
         MyLine1("y2") = ParentY
         MyLine1("Parent") = 0
   Else
         If MyOpr.RecordCount > 0 Then MyOpr.MoveFirst
         MyOpr.Index = "OprIdIx"
         MyOpr.Seek "=", MyTemp
         MyLine1("x2") = MyOpr("OX")
         MyLine1("y2") = MyOpr("OY")
         MyLine1("Parent") = ProcessParent(ParentIndex)
   End If
   MyLine1.Update
   Call drawlines1
   Call drawlines
End Sub

Sub CopyLibProcess ()
   MyProcess.Index = "IdIx"
   MyProcess.Seek "=", ProcessIndex
   IReferTo = MyProcess("ProcessRefer")

   If MyNodes.RecordCount <> 0 Then
         ' delete if any existing nodes
         If MyNodes.RecordCount > 0 Then MyNodes.MoveFirst
         Do Until MyNodes.EOF
            If MyNodes("ProcessId") = ProcessIndex Then
                  MyNodes.Delete
            End If
            If MyNodes.EOF Then Exit Do
            MyNodes.MoveNext
         Loop
   End If

   If MyCon.RecordCount <> 0 Then
         ' delete if any connections
         MyCon.MoveFirst
         Do Until MyCon.EOF
            If MyCon("ProcessId") = ProcessIndex Then
                  MyCon.Delete
            End If
```

```
            If MyCon.EOF Then Exit Do
            MyCon.MoveNext
          Loop
    End If

    LibNodes.Index = "ProcessIdIx"
    If LibNodes.RecordCount > 0 Then If LibNodes.RecordCount > 0 Then LibNodes.MoveFirst
    LibNodes.Seek "=", IReferTo
    If Not LibNodes.NoMatch Then
          If LibNodes.RecordCount > 0 Then If LibNodes.RecordCount > 0 Then
LibNodes.MoveFirst
          Do Until LibNodes.EOF
            If LibNodes("ProcessId") = IReferTo Then
                  TempNodeNum = LibNodes("NodeId")
                  TempNodeType = LibNodes("NodeType")
                  TempNodeDesc = LibNodes("NodeName")
                  TempNodeDur = LibNodes("Duration")
                  TempNodeX = LibNodes("NodeX")
                  TempNodeY = LibNodes("NodeY")
                  TempNodeCon = LibNodes("ConGen")
                  TempNodeGen = LibNodes("GenQue")
                  TempNodeInit = LibNodes("InitQue")
                  MyNodes.AddNew
                  MyNodes("ProcessId") = ProcessIndex
                  MyNodes("NodeId") = TempNodeNum
                  MyNodes("NodeType") = TempNodeType
                  MyNodes("NodeName") = TempNodeDesc
                  MyNodes("Duration") = TempNodeDur
                  MyNodes("NodeX") = TempNodeX
                  MyNodes("NodeY") = TempNodeY
                  MyNodes("ConGen") = TempNodeCon
                  MyNodes("GenQue") = TempNodeGen
                  MyNodes("InitQue") = TempNodeInit
                  MyNodes.Update
            End If
            If LibNodes.EOF Then Exit Do
            LibNodes.MoveNext
          Loop
    End If

    If LibCon.RecordCount > 0 Then
          LibCon.Index = "ProcessIdIx"
          If LibCon.RecordCount > 0 Then LibCon.MoveFirst
          LibCon.Seek "=", IReferTo
          If Not LibCon.NoMatch Then
            Do Until LibCon.EOF
                  If LibCon("ProcessId") = IReferTo Then
                    TempConnectNum = LibCon("ConnectionId")
                    TempPredNode = LibCon("Predecessor")
                    TempSucNode = LibCon("Successor")
                    TempConType = LibCon("ConnectionType")
                    MyCon.AddNew
                    MyCon("ConnectionId") = LibCon("ConnectionId")
                    MyCon("ProcessId") = ProcessIndex
```

```
                    MyCon("Predecessor") = TempPredNode
                    MyCon("Successor") = TempSucNode
                    MyCon("ConnectionType") = TempConType
                    MyCon.Update
                End If
                If LibCon.EOF Then Exit Do
                LibCon.MoveNext
            Loop
        End If
    End If
End Sub

Sub DefMod_Click ()
    ModularOperation = 1
    NewForm.Show
End Sub

Sub DeleteProcess1 ()
        ' if the user wants to delete the process
        ' delete process
        If MyProcess.RecordCount > 0 Then If MyProcess.RecordCount > 0 Then
MyProcess.MoveFirst
        MyProcess.Index = "IdIx"
        MyProcess.Seek "=", ProcessIndex
        MyProcess.Delete
        ' delete its location information
        If MyLine1.RecordCount > 0 Then MyLine1.MoveFirst
        MyLine1.Index = "ChildIx"
        MyLine1.Seek "=", ProcessIndex
        If Not MyLine1.NoMatch Then MyLine1.Delete
        ' delete nodes if existing
        If MyNodes.RecordCount > 0 Then
            MyNodes.MoveFirst
            Do Until MyNodes.EOF
                If MyNodes("ProcessId") = ProcessIndex Then
                    MyNodes.Delete
                End If
                MyNodes.MoveNext
            Loop
        End If
        'delete connection between nodes for that table
        If MyCon.RecordCount > 0 Then
            MyCon.MoveFirst
            Do Until MyCon.EOF
                If MyCon("ProcessId") = ProcessIndex Then
                    MyCon.Delete
                End If
                MyCon.MoveNext
            Loop
        End If
        ' delete process inter-dependency
        If MyProInter.RecordCount > 0 Then
            MyProInter.MoveFirst
            Do Until MyProInter.EOF
```

```
                If MyProInter("PreProcess") = ProcessIndex Then
                  If MyNodes.RecordCount > 0 Then MyNodes.MoveFirst
                  Do Until MyNodes.EOF
                        If MyNodes("ProcessId") = MyProInter("SucProcess") And
MyNodes("NodeId") = MyProInter("SucNodeId") Then
                          MyNodes.Delete
                        End If
                        MyNodes.MoveNext
                  Loop
                  MyProInter.Delete
                End If
                MyProInter.MoveNext
            Loop
          End If
          If MyProInter.RecordCount > 0 Then
            MyProInter.MoveFirst
            Do Until MyProInter.EOF
                If MyProInter("SucProcess") = ProcessIndex Then
                  MyNodes.MoveFirst
                  Do Until MyNodes.EOF
                        If MyNodes("ProcessId") = MyProInter("PreProcess") And
MyNodes("NodeId") = MyProInter("PreNodeId") Then
                          MyNodes.Delete
                        End If
                        MyNodes.MoveNext
                  Loop
                  MyProInter.Delete
                End If
                MyProInter.MoveNext
            Loop
          End If
          ' remove all ArnFrnTable entries ************
          If MyArn.RecordCount > 0 Then
            MyArn.MoveFirst
            Do Until MyArn.EOF
                If MyArn("ProcessId") = ProcessIndex Then
                  MyArn.Delete
                End If
                MyArn.MoveNext
            Loop
          End If

          ' remove all process objects and reload
          ProjForm.MousePointer = 0
          For i = 1 To NProcess%
            Unload Process1(i)
          Next
          ProjForm.Cls
          NProcess% = 0
          If MyProcess.RecordCount = 0 Then Exit Sub
          If MyProcess.RecordCount > 0 Then MyProcess.MoveFirst
          Do Until MyProcess.EOF
            If MyProcess("ProcessType") <> 111 Then
                NProcess% = NProcess% + 1
```

```
                Load Process1(NProcess%)
                Process1(NProcess%).Top = MyProcess("OY")
                Process1(NProcess%).Left = MyProcess("OX")
                Process1(NProcess%).Visible = True
                Process1(NProcess%).Caption = Left$(MyProcess("ProcessName"), 6)
                Process1(NProcess%).ProcessId = MyProcess("ProcessId")
            End If
            If MyProcess.EOF Then Exit Do
            MyProcess.MoveNext
        Loop
End Sub

Sub DelOpr_Click ()

If MyOpr.RecordCount = 0 Then
   MsgBox "No Operations defined for the project.", 48
   Exit Sub
End If
   MsgBox "Double-Click on the operation to delete", 48
   ProjForm.MousePointer = 2
   DeleteOperation = -1
End Sub

Sub DelPro_Click ()
If MyProcess.RecordCount = 0 Then
   MsgBox "No processes defined for the project.", 48
   Exit Sub
End If
ProjForm.MousePointer = 2
DeleteProcess = -1
MDIForm1.Panel3D1.Caption = "Click on a process object."
End Sub

Sub drawlines ()
   Cls
   If MyLine.RecordCount > 0 Then MyLine.MoveFirst
   Do Until MyLine.EOF
         ChildX = MyLine("x1")
         ChildY = MyLine("y1")
         ParentX = MyLine("X2")
         ParentY = MyLine("Y2")
         If ChildY < ParentY Then
            ChildY = ChildY + 23
            ChildX = ChildX + 46
            ParentX = ParentX + 46
            ParentY = ParentY
            ProjForm.Line (ChildX, ChildY)-(ChildX, ChildY + 20)
            ProjForm.Line (ChildX, ChildY + 20)-(ParentX, ChildY + 20)
            ProjForm.Line (ParentX, ChildY + 20)-(ParentX, ParentY)
         Else
            ChildY = ChildY
            ChildX = ChildX + 46
            ParentX = ParentX + 46
            ParentY = ParentY + 23
```

```
            ProjForm.Line (ParentX, ParentY)-(ParentX, ParentY + 20)
            ProjForm.Line (ParentX, ParentY + 20)-(ChildX, ParentY + 20)
            ProjForm.Line (ChildX, ParentY + 20)-(ChildX, ChildY)
         End If
         MyLine.MoveNext
   Loop

End Sub

Sub drawlines1 ()

   If MyLine1.RecordCount > 0 Then MyLine1.MoveFirst
   Do Until MyLine1.EOF
         ChildX = MyLine1("x1")
         ChildY = MyLine1("y1")
         ParentX = MyLine1("X2")
         ParentY = MyLine1("Y2")
         If ChildY < ParentY Then
            ChildY = ChildY + 23
            ChildX = ChildX + 46
            ParentX = ParentX + 46
            ParentY = ParentY
            ProjForm.Line (ChildX, ChildY)-(ChildX, ChildY + 20)
            ProjForm.Line (ChildX, ChildY + 20)-(ParentX, ChildY + 20)
            ProjForm.Line (ParentX, ChildY + 20)-(ParentX, ParentY)
         Else
            ChildY = ChildY
            ChildX = ChildX + 46
            ParentX = ParentX + 46
            ParentY = ParentY + 23
            ProjForm.Line (ParentX, ParentY)-(ParentX, ParentY + 20)
            ProjForm.Line (ParentX, ParentY + 20)-(ChildX, ParentY + 20)
            ProjForm.Line (ChildX, ParentY + 20)-(ChildX, ChildY)
         End If
         MyLine1.MoveNext
   Loop
End Sub

Sub EditOpr_Click ()
   MsgBox "Not implemented in Version 1.0", 48
End Sub

Sub EditPro_Click ()
   MsgBox "Not implemented in Version 1.0", 48
End Sub

Sub EditProcessmenu_Click ()
   If MyProcess.RecordCount = 0 Then
         MsgBox ("No processes defined for this project")
         Exit Sub
   End If
   EditProcess = 1•
   MsgBox ("double click on the process to edit")
```

```
End Sub

Sub ExitMenu_Click ()
    End
End Sub

Sub FileNew_Click ()
        ProjEdit.Show 1
        If ProjEdit.Text1.Text = " " Then Exit Sub
        Temp$ = Left$(ProjEdit.Text1.Text, 15)
        ProjForm.Shape1.Visible = True
        ProjForm.Label1.Caption = Temp$
        ProjForm.Label1.Visible = True
        OprMenu.Enabled = True
        ProNet.Enabled = True•
        PrAdd.Enabled = True
        OprSeq.Enabled = True
        ResPool.Enabled = True
        FileNew.Enabled = False
        EditMenu.Enabled = True
        Image1.Visible = True
        Label2.Visible = True
        EditMenu.Enabled = True
        DelOpr.Enabled = True
        DelPro.Enabled = True
        DefMod.Enabled = True
        RefPro.Enabled = True
        TransMenu.Enabled = True
        Unload ProjEdit
        ProjectNotStarted = -1
End Sub

Sub fitbar ()

If WindowState = 2 Then
  Vscroll1.Left = Abs(scaleleft + scalewidth - 20)
  Vscroll1.Top = scaletop + 5
  Vscroll1.Height = Abs(scaleHeight - 30)

  hscroll1.Top = Abs(scaletop + scaleHeight - 25)
  hscroll1.Left = scaleleft + 5
  hscroll1.Width = Abs(scalewidth - 25)
Else
  Vscroll1.Left = Abs(scaleleft + scalewidth - 20)
  Vscroll1.Top = scaletop + 5
  Vscroll1.Height = Abs(scaleHeight - 30)

  hscroll1.Top = Abs(scaletop + scaleHeight - 25)
  hscroll1.Left = scaleleft + 5
  hscroll1.Width = Abs(scalewidth - 25)
End If

End Sub
```

```
Sub Form_Activate ()
If ProjectNotStarted <> -1 Then Exit Sub
If ScreenSwitch = 1 Then
   If MyOpr.RecordCount > 0 Then
         Call drawlines
   End If
   If MyProcess.RecordCount > 0 Then
         Call drawlines1
   End If
   ScreenSwitch = 0
End If

End Sub

Sub Form_DragDrop (Source As Control, x As Single, y As Single)
If MoveWhat = 1 Then
   Opr1(OprIndex%).Move x, y
   If MyLine.RecordCount > 0 Then MyLine.MoveFirst
   MyLine.Index = "ChildIx"
   Do Until MyLine.EOF
         MyLine.Seek "=", ActualOprIndex%
         If Not MyLine.NoMatch Then
            MyLine.Edit•
            MyLine("x1") = x
            MyLine("y1") = y
            MyLine.Update
            Exit Do
         End If
         MyLine.MoveNext
   Loop
   If MyLine.RecordCount > 0 Then MyLine.MoveFirst
   MyLine.Index = "ParentIx"
   Do Until MyLine.EOF
         If MyLine.EOF Then Exit Do
         If MyLine("Parent") = ActualOprIndex% Then
            MyLine.Edit
            MyLine("x2") = x
            MyLine("y2") = y
            MyLine.Update•
         End If
         MyLine.MoveNext
   Loop
   If MyLine1.RecordCount = 0 Then GoTo Skip1
   MyLine1.MoveFirst
   MyLine1.Index = "ParentIx"
   Do Until MyLine1.EOF
         If MyLine1.EOF Then Exit Do
         If MyLine1("Parent") = ActualOprIndex% Then
            MyLine1.Edit
            MyLine1("x2") = x
            MyLine1("y2") = y
            MyLine1.Update
         End If
         MyLine1.MoveNext
```

```
    Loop
    If MyLine.RecordCount > 0 Then MyLine.MoveFirst
Skip1:
    If MyOpr.RecordCount > 0 Then MyOpr.MoveFirst
    MyOpr.Index = "OprIdIx"
    MyOpr.Seek "=", ActualOprIndex%
    If Not MyOpr.NoMatch Then
            MyOpr.Edit
            MyOpr("OX") = x
            MyOpr("OY") = y
            MyOpr.Update
    End If
    MoveWhat = 0
    Cls
    'If MyOpr.RecordCount > 0 Then MyOpr.MoveFirst
    'Do Until MyOpr.EOF
    '   If MyOpr("OprType") = 1 Then
    '      LeftY = MyOpr("OY") + (29 / 2)
    '      LeftX = MyOpr("OX") - 2
    '      RightY = LeftY
    '      RightX = LeftX + 4 + 67
    '      FillColor = 0
    '      FillStyle = 0
    '      Circle (LeftX, LeftY), 2, RGB(255, 0, 0)
    '      Circle (RightX, RightY), 2, RGB(255, 0, 0)
    '   End If
    '   MyOpr.MoveNext
    'Loop
    'If MyProcess.RecordCount = 0 Then Exit Sub
    'MyProcess.MoveFirst
    'Do Until MyProcess.EOF
    '   If MyProcess("Refer") = -1 Then
    '      LeftY = MyProcess("OY") + (33 / 2)
    '      LeftX = MyProcess("OX") - 2
    '      RightY = LeftY
    '      RightX = LeftX + 4 + 79
    '      FillColor = 0
    '      FillStyle = 0
    '      Circle (LeftX, LeftY), 2, RGB(255, 0, 0)
    '      Circle (RightX, RightY), 2, RGB(255, 0, 0)
    '   End If
    '   MyProcess.MoveNext
    'Loop
ElseIf MoveWhat = 2 Then
    Process1(MyProcessIndex).Move x, y
    If MyLine1.RecordCount > 0 Then MyLine1.MoveFirst
    MyLine1.Index = "ChildIx"
    Do Until MyLine1.EOF
            MyLine1.Seek "=", ProcessIndex
            If Not MyLine1.NoMatch Then
               MyLine1.Edit
               MyLine1("x1") = x
               MyLine1("y1") = y
               MyLine1.Update
```

```
            Exit Do
          End If•
          MyLine1.MoveNext
    Loop
    If MyProcess.RecordCount > 0 Then If MyProcess.RecordCount > 0 Then
MyProcess.MoveFirst
    MyProcess.Index = "IdIx"
    MyProcess.Seek "=", ProcessIndex
    If Not MyProcess.NoMatch Then
          MyProcess.Edit
          MyProcess("OX") = x
          MyProcess("OY") = y
          MyProcess.Update
    End If
    MoveWhat = 0
    Cls

    'If MyProcess.RecordCount > 0 Then If MyProcess.RecordCount > 0 Then
MyProcess.MoveFirst
    'Do Until MyProcess.EOF
    '   If MyProcess("Refer") = -1 Then
    '      LeftY = MyProcess("OY") + (33 / 2)
    '      LeftX = MyProcess("OX") - 2
    '      RightY = LeftY
    '      RightX = LeftX + 4 + 79
    '      FillColor = 0
    '      FillStyle = 0
    '      Circle (LeftX, LeftY), 2, RGB(255, 0, 0)
    '      Circle (RightX, RightY), 2, RGB(255, 0, 0)
    '   End If
    '   MyProcess.MoveNext
    'Loop
    'If MyOpr.RecordCount > 0 Then MyOpr.MoveFirst
    'Do Until MyOpr.EOF
    '   If MyOpr("OprType") = 1 Then
    '      LeftY = MyOpr("OY") + (29 / 2)
    '      LeftX = MyOpr("OX") - 2
    '      RightY = LeftY
    '      RightX = LeftX + 4 + 67
    '      FillColor = 0
    '      FillStyle = 0
    '      Circle (LeftX, LeftY), 2, RGB(255, 0, 0)
    '      Circle (RightX, RightY), 2, RGB(255, 0, 0)
    '   End If
    '   MyOpr.MoveNext
    'Loop
End If
    Call drawlines
    Call drawlines1
End Sub

Sub Form_GotFocus ()
If ProjectNotStarted <> -1 Then Exit Sub
If ScreenSwitch = 1 Then
```

```
    If MyOpr.RecordCount > 0 Then
        Call drawlines
    End If
    If MyProcess.RecordCount > 0 Then
        Call drawlines1
    End If
    ScreenSwitch = 0
End If
End Sub

Sub Form_Load ()
    fitbar
    Call SetDatabase1
End Sub


Sub Form_MouseDown (Button As Integer, Shift As Integer, x As Single, y As Single)

If AddOpr% <> 1 And AddProcess% <> 1 And EditProcess <> 1 Then Exit Sub
    ' adding an operation
    If AddOpr% = 1 Then
        NOpr% = NOpr% + 1
        MDIForm1.Panel3D1.Caption = DefaultCaption$
        OprEdit.Show 1
        If CancelAdd = -1 Then
            CancelAdd = 0
            AddOpr% = 0
            NOpr% = NOpr% - 1
            Exit Sub
        End If
        Load Opr1(NOpr%)
        Opr1(NOpr%).Top = y
        Opr1(NOpr%).Left = x
        Opr1(NOpr%).Visible = True
        Opr1(NOpr%).Caption = Left$(OprEdit.Text1.Text, 10)
        Opr1(NOpr%).Tag = Str$(OprEdit.Option3D1.Value)
        Opr1(NOpr%).OprId = Val(OprEdit.MaskedEdit1.Text)
        If MyOpr.RecordCount > 0 Then MyOpr.MoveFirst
        MyOpr.Index = "OprIdIx"
        MyOpr.Seek "=", Opr1(NOpr%).OprId
        If Not MyOpr.NoMatch Then
            MyOpr.Edit
            MyOpr("OX") = x
            MyOpr("OY") = y
            MyOpr.Update
        End If
        Call ConnectOpr
        'If OprEdit!Option3D3.Value = True Then
        '   LeftY = Opr1(NOpr%).Top + (Opr1(NOpr%).Height / 2)
        '   LeftX = Opr1(NOpr%).Left - 2
        '   RightY = LeftY
        '   RightX = LeftX + 4 + Opr1(NOpr%).Width
        '   FillColor = 0
```

```
            '   FillStyle = 0
            '   Circle (LeftX, LeftY), 2, RGB(255, 0, 0)
            '   Circle (RightX, RightY), 2, RGB(255, 0, 0)
            'End If
            AddOpr% = 0
            Unload OprEdit
        End If
    ' adding a process

        If AddProcess% = 1 Then
            If MyOpr.RecordCount = 0 Then
                MsgBox ("At least one Operation must exit to add a process")
                AddProcess = 0
                Exit Sub
            End If
            NProcess% = NProcess% + 1
            MDIForm1.Panel3D1.Caption = DefaultCaption$
            ProcessForm.Show 1
            If CancelAdd = -1 Then
                CancelAdd = 0
                NProcess% = NProcess% - 1
                AddProcess% = 0
                Exit Sub
            End If
            Load Process1(NProcess%)
            Process1(NProcess%).Top = y
            Process1(NProcess%).Left = x
            Process1(NProcess%).Visible = True
            Process1(NProcess%).Caption = Left$(ProcessForm.Text1.Text, 10)
            Process1(NProcess%).ProcessId = Val(ProcessForm.MaskedEdit1.Text)
            Process1(NProcess%).ProcessName = ProcessForm.Text1.Text
            'If ProcessForm!Option3D1.Value = -1 Then
            '   Process1(NProcess%).Caption = "Mod_" + ProcessForm.Text1.Text
            '   Call CopyLibProcess
            'End If
            If MyProcess.RecordCount > 0 Then MyProcess.MoveFirst
            MyProcess.Index = "IdIx"
            MyProcess.Seek "=", Process1(NProcess%).ProcessId
            If Not MyProcess.NoMatch Then
                MyProcess.Edit
                MyProcess("OX") = x
                MyProcess("OY") = y
                MyProcess.Update
            End If
            Call ConnectProcess
            AddProcess% = 0
            Unload ProcessForm
        End If
    MousePointer = 0
    End Sub

    Sub Form_MouseMove (Button As Integer, Shift As Integer, x As Single, y As Single)
        Label3.Caption = ""
        Label3.Visible = False
```

```
End Sub

Sub Form_Resize ()
   fitbar
End Sub

Sub HelpMeu_Click ()
   AboutBox.Show 1
End Sub

Sub HScroll1_Change ()
   Call StoreOld
   Cls
   scaleleft = hscroll1.Value * Opr1(0).Width

   Shape1.Top = 28
   Shape1.Left = 317
   Label1.Top = 42
   Label1.Left = 360
   If NOpr% = 0 Then Exit Sub
   For i = 1 To NOpr%
                  Opr1(i).Top = oldyop(i)
                  Opr1(i).Left = oldxop(i)
   Next i
   For i = 1 To NProcess%
         Process1(i).Top = oldypr(i)
         Process1(i).Left = oldxpr(i)
   Next i
   Call drawlines
   Call drawlines1
End Sub

Sub Image1_Click ()
ResForm.Show
End Sub

Sub Label2_Click ()
'resource.Show 1
End Sub

Sub List1_DblClick ()
ProcessIndex = ModOprProcessIndex(List1.ListIndex)
AddPro% = 0
List1.Visible = False
ProjForm.WindowState = 1
Load Cyclone
End Sub

Sub LoadProject ()

Cls
For i = 1 To NOpr%
   Unload Opr1(i)
Next i
```

```
For i = 1 To NProcess%
   Unload Process1(i)
Next i

NOpr% = 0
NProcess% = 0

Call SetDatabase

Temp$ = FilePathName$
Do While InStr(1, Temp$, "\")
   Position% = InStr(1, Temp$, "\")
   Temp$ = Right$(Temp$, Len(Temp$) - Position%)
Loop
Temp$ = Left$(Temp$, Len(Temp$) - 4)
   If IndexO = 0 Then
         ProjForm.Shape1.Visible = True
         ProjForm.Label1.Caption = Temp$
         ProjForm.Label1.Visible = True
         OprMenu.Enabled = True
         ProNet.Enabled = True
         If MyOpr.RecordCount > 0 Then MyOpr.MoveFirst
         Do Until MyOpr.EOF
            If MyOpr("OprType") <> 111 Then
                  NOpr% = NOpr% + 1
                  Load Opr1(NOpr%)
                  Opr1(NOpr%).Top = MyOpr("OY")
                  Opr1(NOpr%).Left = MyOpr("OX")
                  Opr1(NOpr%).Visible = True
                  Opr1(NOpr%).Caption = Left$(MyOpr("OpName"), 10)•
                  Opr1(NOpr%).Tag = MyOpr("Lowest")
                  Opr1(NOpr%).OprId = MyOpr("OprId")
                  'If MyOpr("OprType") = 1 Then
                  '   Opr1(NOpr%).Caption = "Mod_" + MyOpr("OpName")
                  'End If
            End If
            If MyOpr.EOF Then Exit Do
            MyOpr.MoveNext
         Loop
         If MyLine.RecordCount <> 0 Then Call drawlines
         'If MyProcess.RecordCount = 0 Then Exit Sub
         If MyProcess.RecordCount > 0 Then MyProcess.MoveFirst
         Do Until MyProcess.EOF
            If MyProcess("ProcessType") <> 111 Then
                  NProcess% = NProcess% + 1
                  Load Process1(NProcess%)
                  Process1(NProcess%).Top = MyProcess("OY")
                  Process1(NProcess%).Left = MyProcess("OX")
                  Process1(NProcess%).Visible = True
                  Process1(NProcess%).Caption = Left$(MyProcess("ProcessName"), 10)
                  Process1(NProcess%).ProcessId = MyProcess("ProcessId")
                  Process1(NProcess%).ProcessName = MyProcess("ProcessName")
                  If MyProcess("Refer") = -1 Then
                     Process1(NProcess%).Tag = "R"
```

```
                '   Process1(NProcess%).Caption = "Mod_" + MyProcess("ProcessName")
                End If
            End If
            MyProcess.MoveNext
            If MyProcess.EOF Then Exit Do
        Loop
        If MyLine1.RecordCount <> 0 Then Call drawlines1
    Else
        MsgBox "Project already open...", 48
        Exit Sub
    End If
    IndexO = 1

End Sub


Sub ModToWBS_Click ()
    MyGroup = 0: ModNOpr% = 0: ModNProcess% = 0: ModularOperation = 0
    Unload NewForm
    ScreenSwitch = 1
End Sub

Sub OpenFileMenu_Click ()
    CMDialog1.Filter = "Project files (*.sim)|*.sim"
    CMDialog1.FilterIndex = 1
    CMDialog1.DefaultExt = "SIM"
    CMDialog1.Action = 1
    FilePathName$ = CMDialog1.Filename
    OpenProject = -1
    If FilePathName$ = "" Then
        OpenProject = 0•
        Exit Sub
    End If
    OprMenu.Enabled = True
    ProNet.Enabled = True
    PrAdd.Enabled = True
    FileNew.Enabled = False
    OprSeq.Enabled = True
    ResPool.Enabled = True
    Image1.Visible = True
    Label2.Visible = True
    RefWin.Enabled = True
    EditMenu.Enabled = True
    EditMenu.Enabled = True
    DelOpr.Enabled = True
    DelPro.Enabled = True
    DefMod.Enabled = True
    RefPro.Enabled = True
    TransMenu.Enabled = True
    EditProcessMenu.Enabled = True
    EditProcess = 0
    ProjForm.MousePointer = 11
    ProjectNotStarted = -1
    Call LoadProject
```

```
    ProjForm.MousePointer = 0
End Sub

Sub Opr1_DblClick (Index As Integer)

If Index <> 0 And DeleteOperation <> -1 And AddPro% = 1 Then
    If MyOpr.RecordCount > 0 Then MyOpr.MoveFirst
    MyOpr.Index = "OprIdIx"
    MyOpr.Seek "=", ActualOprIndex%
    k = 0
    If MyOpr("OprType") = 1 Then
        If MyProcess.RecordCount > 0 Then MyProcess.MoveFirst
        List1.Clear
        Do Until MyProcess.EOF
            If MyProcess("Group") = ActualOprIndex% Then
                List1.AddItem MyProcess("ProcessName")
                ModOprProcessIndex(k) = MyProcess("ProcessId")
                k = k + 1
            End If
            MyProcess.MoveNext
        Loop
        List1.ListIndex = 0
        List1.Left = Opr1(Index).Left + Opr1(Index).Width
        List1.Top = Opr1(Index).Top + Opr1(Index).Height
        List1.Width = 150
        List1.Height = 100
        List1.Visible = True
    End If
End If

If Index <> 0 And DeleteOperation = -1 Then
    DeleteOperation = 0
    Msg$ = "      The selected Operation object will be deleted." & Chr$(10) + Chr$(13)
    Msg$ = Msg$ + " This can effect WBS, operation sequencing, process model" & Chr$(10) +
Chr$(13)
    Msg$ = Msg$ + "                and resource allocation." & Chr$(10) + Chr$(13)
    Msg$ = Msg$ + "              Do you want to continue?"
    Response = MsgBox(Msg$, 36)
    If Response = 7 Then
        ProjForm.MousePointer = 0
        Exit Sub
    End If
    DeleteOperation = 0
    ' if the user wants to delete the operation
    ' delete operation from oprtable
    If MyOpr.RecordCount > 0 Then MyOpr.MoveFirst
    MyOpr.Index = "OprIdIx"
    MyOpr.Seek "=", ActualOprIndex%
    MyOpr.Delete
    ' delete the Operation Sequencing
    If MyRel.RecordCount = 0 Then GoTo Jump3
    MyRel.MoveFirst
    Do Until MyRel.EOF
        If MyRel("Parent") = ActualOprIndex% Then
```

```
            MyRel.Delete
        End If
        If MyRel.EOF Then Exit Do
        MyRel.MoveNext
    Loop
    If MyRel.RecordCount = 0 Then GoTo Jump3
    MyRel.MoveFirst
    Do Until MyRel.EOF
        If MyRel("Predecessor") = ActualOprIndex% Or MyRel("Successor") = ActualOprIndex%
Then
            MyRel.Delete
        End If
        If MyRel.EOF Then Exit Do
        MyRel.MoveNext
    Loop

Jump3:
    ' delete its location information
    If MyLine.RecordCount > 0 Then MyLine.MoveFirst
    MyLine.Index = "ChildIx"
    MyLine.Seek "=", ActualOprIndex%
    MyLine.Delete
    ' all children below the operation have to be deleted
    If MyOpr.RecordCount = 0 Then GoTo Jump1
    MyOpr.MoveFirst
    MyOpr.Index = "OprIdIx"
    Do Until MyOpr.EOF
        TestLine = MyOpr("OprId")
        TestParent = MyOpr("ParentId")
        If TestParent = 0 Then GoTo Jump4
        MyOpr.Seek "=", TestParent
        If MyOpr.NoMatch Then
            MyOpr.Seek "=", TestLine
            MyOpr.Delete
            ' delete its location information
            If MyLine.RecordCount > 0 Then MyLine.MoveFirst
            MyLine.Index = "ChildIx"
            MyLine.Seek "=", TestLine
            If Not MyLine.NoMatch Then MyLine.Delete
            If MyRel.RecordCount <> 0 Then
                MyRel.MoveFirst
                Do Until MyRel.EOF
                    If MyRel("Parent") = TestLine Then
                        MyRel.Delete
                    End If
                    If MyRel.EOF Then Exit Do
                    MyRel.MoveNext
                Loop
            End If
            If MyRel.RecordCount <> 0 Then
                MyRel.MoveFirst
                Do Until MyRel.EOF
                    If MyRel("Predecessor") = TestLine Or MyRel("Successor") = TestLine Then
                        MyRel.Delete
```

```
                End If
                If MyRel.EOF Then Exit Do
                MyRel.MoveNext
              Loop
          End If
        Else
          MyOpr.Seek "=", TestLine
        End If
Jump4:
        If MyOpr.EOF Then Exit Do
        MyOpr.MoveNext
  Loop
  ' delete any connected process
Jump1:
  If MyProcess.RecordCount = 0 Then GoTo Jump2
  If MyProcess.RecordCount > 0 Then MyProcess.MoveFirst
  Do Until MyProcess.EOF
        TestProcessParent = MyProcess("ParentId")
        If MyOpr.RecordCount > 0 Then MyOpr.MoveFirst
        MyOpr.Index = "OprIdIx"
        MyOpr.Seek "=", TestProcessParent
        If MyOpr.NoMatch Then
          ProcessIndex = MyProcess("ProcessId")
          Call DeleteProcess1
          If MyProcess.RecordCount = 0 Then Exit Do
          MyProcess.MoveFirst
        Else
          If MyProcess.EOF Then Exit Do
          MyProcess.MoveNext
        End If
  Loop
Jump2:
        ' remove all process objects and reload
        ProjForm.MousePointer = 0
        For i = 1 To NOpr%
          Unload Opr1(i)
        Next
        ProjForm.Cls
        NOpr% = 0
        If MyOpr.RecordCount = 0 Then Exit Sub
        MyOpr.MoveFirst
        Do Until MyOpr.EOF
          If MyOpr("OprType") <> 111 Then
                NOpr% = NOpr% + 1
                Load Opr1(NOpr%)
                Opr1(NOpr%).Top = MyOpr("OY")
                Opr1(NOpr%).Left = MyOpr("OX")
                Opr1(NOpr%).Visible = True
                Opr1(NOpr%).Caption = Left$(MyOpr("OpName"), 10)
                Opr1(NOpr%).OprId = MyOpr("OprId")
          End If
          If MyOpr.EOF Then Exit Do
          MyOpr.MoveNext•
        Loop
```

```
        If MyLine.RecordCount <> 0 Then Call drawlines
        If MyLine1.RecordCount <> 0 Then Call drawlines1
End If
End Sub

Sub Opr1_MouseMove (Index As Integer, Button As Integer, Shift As Integer, x As Single, y As
Single)
   OprIndex% = Index
   ActualOprIndex% = Opr1(Index).OprId
   MoveWhat = 1
   'Label3.Caption = Opr1(Index).Caption
   'Label3.Visible = True
   'Label3.AutoSize = True
   'Label3.Top = Opr1(Index).Top + Opr1(Index).Height
   'Label3.Left = Opr1(Index).Left + Opr1(Index).Width / 2
End Sub

Sub OprMenu_Click ()

   ProjForm.MousePointer = 2
   MDIForm1.Panel3D1.Caption = "Click on the form to place the operation object."
   AddOpr% = 1
   AddProcess% = 0
   EditProcess = 0
   RefWin.Enabled = True
End Sub

Sub OprSeq_Click ()
   OSM = 1
   OprIndex% = 0
   ProcessIndex = 0
   oprseqf.Show
End Sub

Sub PrAdd_Click ()
   ProjForm.MousePointer = 2
   MDIForm1.Panel3D1.Caption = "Click on the form to place the process object."
   AddProcess% = 1
   AddOpr% = 0
   EditProcess = 0
   RefWin.Enabled = True
End Sub

Sub Prnt_Click ()
Call PrintData
End Sub

Sub Process1_DblClick (Index As Integer)

   If Index <> 0 And AddPro% = 1 Then
         AddPro% = 0
         ProjForm.WindowState = 1
         Load Cyclone
   End If
```

```
  If Index <> 0 And DeleteProcess = -1 Then
        DeleteProcess = 0
        Msg$ = "      The selected  PROCESS object will be deleted." & Chr$(10) + Chr$(13)
        Msg$ = Msg$ + " This can effect process model, process inter-dependency" & Chr$(10)
+ Chr$(13)
        Msg$ = Msg$ + "                  and resource allocation." & Chr$(10) + Chr$(13)
        Msg$ = Msg$ + "                Do you want to continue?"
        Response = MsgBox(Msg$, 36)
        If Response = 7 Then
          ProjForm.MousePointer = 0
          MDIForm1.Panel3D1.Caption = DefaultCaption$
          Exit Sub
        End If
        Call DeleteProcess1
        If MyLine.RecordCount <> 0 Then Call drawlines
        If MyLine1.RecordCount <> 0 Then Call drawlines1
  End If
  If Index <> 0 And EditProcess = 1 Then
        ProcessForm.MaskedEdit1.Enabled = False
        ProcessForm.Combo1.Enabled = True
        ProcessForm.Combo2.Enabled = False
        ProcessForm.Combo2.Visible = False
        ProcessForm.Frame3D1.Visible = False
        ProcessForm.Label5.Visible = False
        ProcessForm.Label6.Visible = False
        ProcessForm.Option3D1.Visible = False
        ProcessForm.Option3D2.Visible = False
        ProcessForm.Show 1
        Call drawlines
        Call drawlines1
  End If
End Sub

Sub Process1_MouseMove (Index As Integer, Button As Integer, Shift As Integer, x As Single, y
As Single)
  ProcessIndex = Process1(Index).ProcessId
  MyProcessIndex = Index
  MoveWhat = 2
  'Label3.Caption = Process1(Index).ProcessName
  'Label3.Visible = True
  'Label3.AutoSize = True
  'Label3.Top = Process1(Index).Top + Process1(Index).Height
  'Label3.Left = Process1(Index).Left + Process1(Index).Width / 2
End Sub

Sub ProNet_Click ()
  If MyProcess.RecordCount = 0 Then
        MsgBox "No Processes defined for the project.", 48
        Exit Sub
  End If
  MsgBox "Double click on a Process for which process model is required.", 48
  MousePointer = 2
  AddPro% = 1
  RefWin.Enabled = True
```

```
End Sub

Sub RefPro_Click ()
   ModularProcess = 1
   Cyclone.Show
End Sub


Sub RefWin_Click ()
   If MyLine.RecordCount <> 0 Then Call drawlines
   If MyLine1.RecordCount <> 0 Then Call drawlines1
End Sub

Sub ResPool_Click ()
ResForm.Show
End Sub

Sub StoreOld ()

   For i = 1 To NOpr%
         oldyop(i) = Opr1(i).Top
         oldxop(i) = Opr1(i).Left
   Next i
   For i = 1 To NProcess%
         oldypr(i) = Process1(i).Top
         oldxpr(i) = Process1(i).Left
   Next i

End Sub

Sub TransMenu_Click ()

Msg$ = "Tranlation should be performed after completion of the" & Chr$(10) + Chr$(13)
Msg$ = Msg$ + "              model definition for the project." & Chr$(10) + Chr$(13)
Msg$ = Msg$ + "              Do you want to continue?"
Response = MsgBox(Msg$, 36)
If Response = 7 Then
   Exit Sub
End If
ProjForm.WindowState = 1
TranForm.Show
End Sub

Sub VScroll1_Change ()
   Call StoreOld
   Cls
   scaletop = Vscroll1.Value * Opr1(0).Height

   Shape1.Top = 28
   Shape1.Left = 317
   Label1.Top = 42
   Label1.Left = 360

   If NOpr% = 0 Then Exit Sub
```

```
  For i = 1 To NOpr%
                Opr1(i).Top = oldyop(i)
                Opr1(i).Left = oldxop(i)
  Next i
  For i = 1 To NProcess%
        Process1(i).Top = oldypr(i)
        Process1(i).Left = oldxpr(i)
  Next i


  Call drawlines
  Call drawlines1
End Sub

Sub WBS_Fit_Click ()
  If NOpr% = 0 Then Exit Sub
  WBSFitWindow.Show
End Sub
```

Structure of the project database

AllocateTable

Field Name	*Field Type*	*Remarks*
OprId	Integer	Operation ID
ResourceId	Integer	Resource ID
Quantity	Single	Quantity of the resource allocated.
Priority	Integer	Priority attached to the operation

ArnFrnTable

Field Name	*Field Type*	*Remarks*
ProcessId	Integer	Operation ID
ResourceId	Integer	Resource ID
NodeId	Integer	Node ID
NodeType	Integer	1 for Allocate and 2 for Free Node
Quantity	Single	Quantity of the resource allocated.
Priority	Integer	Priority attached to the operation

ConnectionTable

Field Name	*Field Type*	*Remarks*
ProcessId	Integer	Process ID
ConnectionId	Integer	Connection ID
Predecessor	Integer	Id of the predecessor node
Successor	Integer	Id of the successor node

CrewTable

Field Name	*Field Type*	*Remarks*
CrewId	Integer	Crew combination id
CrewName	Text * 20	Crew combination name
R1	Text * 20	Resource 1
R2	Text * 20	Resource 2
	Text * 20	
R11	Text * 20	Resource 11

FinalRelation

Field Name	*Field Type*	*Remarks*
Predecessor	Integer	Predecessor operation ID
Successor	Integer	Successor operation ID
Parent	Integer	ID of the parent operation
Type	Integer	Type of link 1-serial, 2-parallel, 3-cyclic and 4-hammock
ID	Integer	Relation ID

FinishTable

Field Name	*Field Type*	*Remarks*
OprId	Integer	ID of the Operation
Finish	Integer	Finish operation in terms of the lowest level.

FollowTable

Field Name	*Field Type*	*Remarks*
PreProcess	Integer	ID of the predecessor process
SucProcess	Integer	ID of the successor process
PreNodeID	Integer	ID of the predecessor node
SucNodeID	Integer	ID of the successor node
Quantity	Single	Release quantity

NodeTable

Field Name	*Field Type*	*Remarks*
ProcessId	Integer	Process ID
NodeID	Integer	Node ID
NodeName	Text * 20	Name of the node
NodeType	Integer	1-QUE, 2-COMBI, 3-NORM, 4-CONSOLIDATE, 5-PREDECESSOR, 6-SUCCESSOR, 7-ALLOCATE, 8-FREE
DurationD	Integer	Relation ID
NodeX	Single	Location of the node along the X axis
NodeY	Single	Location of the node along the Y axis
ConGen	Integer	Consolidate quantity
GenQue	Integer	Generate quantity
InitQue	Integer	Initialization quantity for a QUE
Prob	Text * 40	Probability associated with COMBI or NORM

OprTable

Field Name	*Field Type*	*Remarks*
OprId	Integer	Operation ID
Code	Integer	Internal code
OpName	Text * 20	Name of the operation
ParentId	Integer	ID of the parent operation
Lowest	Boolean	True or False
OX	Single	Location along the X axis
OY	Single	Location along the Y axis
ParentName	Text * 20	Name of the parent operation
OprType	Integer	1- modular, 111-child of a modular, 2-simple
Group	Integer	Only for modular
Level	Integer	Level in the hierarchy

Position

Field Name	*Field Type*	*Remarks*
X1	Single	Location of the child operation along the X axis
X2	Single	Location of the parent operation along the X axis
Y1	Single	Location of the child operation along the Y axis
Y2	Single	Location of the parent operation along the Y axis
Parent	Integer	ID of the parent operation
Child	Integer	ID of the Child operation

Position1

Field Name	*Field Type*	*Remarks*
X1	Single	Location of the child process along the X axis
X2	Single	Location of the parent operation along the X axis
Y1	Single	Location of the child process along the Y axis
Y2	Single	Location of the parent operation along the Y axis
Parent	Integer	ID of the parent operation
Child	Integer	ID of the Child process

ProcessTable

Field Name	*Field Type*	*Remarks*
ProcessId	Integer	Process ID
Code	Integer	Internal code
ProcessName	Text * 20	Name of the Process
ParentId	Integer	ID of the parent operation
OX	Single	Location along the X axis
OY	Single	Location along the Y axis
Refer	Integer	1- if refers to modular process, 2- - no reference
ProcessRefer	Integer	ID of the modular process referred
Group	Integer	Only for modular processes

RelationTable

Field Name	*Field Type*	*Remarks*
RelationID	Integer	Relation ID
Predecessor	Integer	ID of the predecessor process
Successor	Integer	ID of the successor process
Type	Integer	1-serial, 2-parallel, 3-cyclic, 4-hammock
Lead	Single	Lead in days
Counter	Integer	For cyclic links
Parent	Integer	ID of the parent operation

ResourceTable

Field Name	*Field Type*	*Remarks*
ResourceID	Integer	Resource ID
ResourceName	Text * 20	Name of the resource
Quantity	Single	Available quantity for the project
Fcost	Currency	Fixed cost per hour
Vcost	Currency	Variable cost per hour

StartTable

Field Name	*Field Type*	*Remarks*
OprId	Integer	ID of the Operation
Start	Integer	Start operation ID in terms of the lowest level.

Appendix B: Details of translation module

In this appendix the Visual Basic program written for the automatic translation of the HSM models into SLAMSYSTEM format is attached.

Program code for translation module

```
VERSION 2.00
Begin Form TranForm
  Caption       = "Translation Module"
  ClientHeight  = 5820
  ClientLeft    = 780
  ClientTop     = 1710
  ClientWidth   = 7365
  Height        = 6510
  Left          = 720
  LinkTopic     = "Form1"
  MDIChild      = -1 'True
  ScaleHeight   = 5820
  ScaleWidth    = 7365
  Top           = 1080
  Width         = 7485
  WindowState   = 2 'Maximized
  Begin TextBox Text1
    Height        = 6030
    Left          = 1335
    MultiLine     = -1 'True
    ScrollBars    = 3 'Both
    TabIndex      = 2
    Top           = 780
    Visible       = 0 'False
    Width         = 9390
  End
  Begin SSFrame Frame3D1
    Alignment     = 2 'Center
    Caption       = "Project Translation Progress"•
    Font3D        = 2 'Raised w/heavy shading
    FontBold      = -1 'True
    FontItalic    = 0 'False
    FontName      = "Times New Roman"
    FontSize      = 12
    FontStrikethru = 0 'False
    FontUnderline = 0 'False
    ForeColor     = &H00FF0000&
    Height        = 2040
    Left          = 3165
    ShadowColor   = 1 'Black
    TabIndex      = 0
    Top           = 2535
    Visible       = 0 'False
    Width         = 5250
    Begin SSPanel Panel3D1
      BackColor     = &H00C0C0C0&
      BevelInner    = 1 'Inset
      BevelOuter    = 1 'Inset
      BevelWidth    = 2
      FloodType     = 1 'Left To Right
      Font3D        = 0 'None
```

```
    ForeColor      =  &H00000000&
    Height         =  465
    Left           =  930
    TabIndex       =  1
    Top            =  810
    Width          =  3570
  End
End
Begin Label Label1
  AutoSize       =  -1  'True
  Caption        =  "Label1"
  FontBold       =  -1  'True
  FontItalic     =  0   'False
  FontName       =  "Times New Roman"
  FontSize       =  12
  FontStrikethru =  0   'False
  FontUnderline  =  0   'False
  Height         =  285
  Left           =  3645
  TabIndex       =  3
  Top            =  3780
  Visible        =  0   'False
  Width          =  690
End
Begin Menu StartTran
  Caption        =  "Translate"
End
Begin Menu DispMenu
  Caption        =  "Display"
  Begin Menu DisOprLink
    Caption      =  "Operation Links"
  End
  Begin Menu DisProLink
    Caption      =  "Process Links"
  End
  Begin Menu ResList
    Caption      =  "Resource List"
  End
  Begin Menu ProTranMod
    Caption      =  "Process Models"
  End
End
Begin Menu WinTrans
  Caption        =  "Window"
  Begin Menu WBSTran
    Caption      =  "WBS"
  End
End
End

Sub DisOprLink_Click ()
  TranForm!Label1.Caption = ""
  TranForm!Label1.Visible = False
  Call DisOprRel
```

```
End Sub

Sub DisProLink_Click ()
   TranForm!Label1.Caption = ""
   TranForm!Label1.Visible = False
   Call DisOprRel
End Sub

Sub ProTranMod_Click ()
   TranForm!Label1.Caption = ""
   TranForm!Label1.Visible = False

      Msg$ = "Individual processes will be translated into SLAM models." & Chr$(10) + Chr$(13)
      Msg$ = Msg$ & "      Run SLAMSYSTEM to simulate the project."
      MsgBox Msg$, 48

End Sub

Sub ResList_Click ()
   TranForm!Label1.Caption = ""
   TranForm!Label1.Visible = False
   Call ResourceList
End Sub

Sub StartTran_Click ()
   Call BackupFile
End Sub

Sub WBSTran_Click ()
   Unload TranForm
   ScreenSwitch = 1
   ProjForm.WindowState = 2
   ProjForm.Show
End Sub

Dim WorkDb As database
Dim WorkRes As Table
Dim WorkRel As Table
Dim WorkProcess As Table
Dim WorkLine As Table
Dim WorkLine1 As Table
Dim WorkOpr As Table
Dim WorkNodes As Table
Dim WorkCon As Table
Dim WorkProInter As Table
Dim WorkCrew As Table
Dim WorkAllocate As Table
Dim WorkArn As Table
Dim WorkFinal As Table
Dim WorkStart As Table
Dim WorkFinish As Table

Dim ChildOpr(50)
Dim StatusOfChild As Integer
```

```
Dim TempOprId As Integer
Dim CheckStart As Integer
Dim CheckFinish As Integer

Sub AnalyzeChild (ChildOpr(), ChildOprCount, StatusOfChild)

For i = 1 To ChildOprCount
   MyOpr.MoveFirst
   Do Until MyOpr.EOF
      If MyOpr("OprId") = ChildOpr(i) Then
         If MyOpr("Lowest") = True And MyOpr("OprType") <> 1 Then
            Exit Do
         Else
            StatusOfChild = -1 ' not all the children are lowest
            Exit Sub
         End If
      End If
      MyOpr.MoveNext
   Loop
   StatusOfChild = 1
Next
End Sub

Sub AnalyzeNotLow (ChildOpr(), ChildOprCount)

' first clear the working database
If WorkStart.RecordCount > 0 Then WorkStart.MoveFirst
Do Until WorkStart.EOF
   WorkStart.Delete
   WorkStart.MoveNext
Loop
If WorkFinish.RecordCount > 0 Then WorkFinish.MoveFirst
Do Until WorkFinish.EOF
   WorkFinish.Delete
   WorkFinish.MoveNext
Loop

If ChildOprCount = 1 Then
   ' check if the child is lowest
   MyOpr.MoveFirst
   Do Until MyOpr.EOF
      If MyOpr("OprId") = ChildOpr(i) Then
         ' if the only child operation is lowest
         If MyOpr("Lowest") = True And MyOpr("OprType") <> 1 Then
            MyStart.AddNew
            MyStart("OprId") = TempOprId
            MyStart("Start") = ChildOpr(1)
            MyStart.Update
            MyFinish.AddNew
            MyFinish("OprId") = TempOprId
            MyFinish("Finish") = ChildOpr(1)
            MyFinish.Update
            Exit Do
         Else
```

```
        ' if the child is not lowest then determine the start and finish
          If MyStart.RecordCount > 0 Then MyStart.MoveFirst
          Do Until MyStart.EOF
            If MyStart("OprId") = ChildOpr(1) Then
              WorkStart.AddNew
              WorkStart("OprId") = MyStart("OprId")
              WorkStart("Start") = MyStart("Start")
              WorkStart.Update
            End If
            MyStart.MoveNext
          Loop
          If WorkStart.RecordCount > 0 Then WorkStart.MoveFirst
          Do Until WorkStart.EOF
            MyStart.AddNew
            MyStart("OprId") = TempOprId
            MyStart("Start") = WorkStart("Start")
            MyStart.Update
            WorkStart.MoveNext
          Loop

          If MyFinish.RecordCount > 0 Then MyFinish.MoveFirst
          Do Until MyFinish.EOF
            If MyFinish("OprId") = ChildOpr(1) Then
              WorkFinish.AddNew
              WorkFinish("OprId") = MyFinish("OprId")
              WorkFinish("Finish") = MyFinish("Finish")
              WorkFinish.Update
            End If
            MyFinish.MoveNext
          Loop
          If WorkFinish.RecordCount > 0 Then WorkFinish.MoveFirst
          Do Until WorkFinish.EOF
            MyFinish.AddNew
            MyFinish("OprId") = TempOprId•
            MyFinish("Finish") = WorkFinish("Finish")
            MyFinish.Update
            WorkFinish.MoveNext
          Loop
        End If
      End If
      MyOpr.MoveNext
    Loop
  Else ' more than one children
  ' clear the working database
    If WorkFinal.RecordCount > 0 Then WorkFinal.MoveFirst
    Do Until WorkFinal.EOF
      WorkFinal.Delete
      WorkFinal.MoveNext
    Loop

    ' collect all the sequencing for the current parent
    For i = 1 To ChildOprCount
      MyRel.MoveFirst
      Do Until MyRel.EOF
```

```
        If MyRel("Predecessor") = ChildOpr(i) Then
          WorkFinal.AddNew
          WorkFinal("Predecessor") = MyRel("Predecessor")
          WorkFinal("Successor") = MyRel("Successor")
          WorkFinal("Type") = MyRel("Type")
          WorkFinal("Parent") = MyRel("Parent")
          WorkFinal("ID") = MyRel("RelationId")
          WorkFinal.Update
        End If•
        MyRel.MoveNext
      Loop
      MyRel.MoveFirst
      Do Until MyRel.EOF
        If MyRel("Successor") = ChildOpr(i) Then
          WorkFinal.AddNew
          WorkFinal("Predecessor") = MyRel("Predecessor")
          WorkFinal("Successor") = MyRel("Successor")
          WorkFinal("Type") = MyRel("Type")
          WorkFinal("Parent") = MyRel("Parent")
          WorkFinal("ID") = MyRel("RelationId")
          WorkFinal.Update
        End If
        MyRel.MoveNext
      Loop
    Next

    ' clear all duplicate relations
    If WorkFinal.RecordCount < 2 Then GoTo Try121
    WorkFinal.Index = "IDIX"
    WorkFinal.MoveFirst
    Do Until WorkFinal.EOF
      Temp1 = WorkFinal("ID")
Try111:
      If WorkFinal.RecordCount >= 2 Then
        WorkFinal.MoveNext
      Else
        Exit Do
      End If
      If WorkFinal("ID") = Temp1 Then
        WorkFinal.Delete
        WorkFinal.MoveNext
        If WorkFinal.EOF Then
          Exit Do
        Else
          WorkFinal.MovePrevious
        End If
        GoTo Try111:
      Else
        WorkFinal.Seek "=", Temp1
        WorkFinal.MoveNext
      End If
    Loop

Try121:
```

```
' check if there is only one relation
WorkFinal.MoveFirst
If WorkFinal.RecordCount = 1 Then
   If WorkFinal("Type") = 1 Then
      CheckStart = WorkFinal("Predecessor")
      Call DeriveStart
      CheckFinish = WorkFinal("Successor")
      Call DeriveFinish
   ElseIf WorkFinal("Type") = 2 Or WorkFinal("Type") = 4 Then
      CheckStart = WorkFinal("Predecessor")
      Call DeriveStart
      CheckStart = WorkFinal("Successor")
      Call DeriveStart
      CheckFinish = WorkFinal("Predecessor")
      Call DeriveFinish
      CheckFinish = WorkFinal("Successor")
      Call DeriveFinish
   ElseIf WorkFinal("Type") = 3 Then
      ' there is a user mistake in the operation sequencing
   End If
Else ' case when more than one relation
   ' first check all the hammock operations
   WorkFinal.MoveFirst
   Do Until WorkFinal.EOF
      If WorkFinal("Type") = 4 Then
         CheckStart = WorkFinal("Predecessor")
         Call DeriveStart
         CheckFinish = WorkFinal("Successor")
         Call DeriveFinish
      End If
      WorkFinal.MoveNext
   Loop

   ' check if there are operations that are not successors to get the start
   For i = 1 To ChildOprCount
      StartOpr = 0
      StartCount = 0
      WorkFinal.MoveFirst
      Do Until WorkFinal.EOF
         If ChildOpr(i) = WorkFinal("Successor") Then
            StartOpr = -1
            StartCount = StartCount + 1
         End If
         WorkFinal.MoveNext
      Loop
      If StartOpr <> -1 Then
         CheckStart = ChildOpr(i)
         Call DeriveStart
      ElseIf StartOpr = -1 Then  ' is a successor but is cyclic
         If StartCount = 1 Then
            WorkFinal.MoveFirst
            Do Until WorkFinal.EOF
               If ChildOpr(i) = WorkFinal("Successor") Then
                  If WorkFinal("Type") = 3 Then
```

```
                    CheckStart = ChildOpr(i)
                    Call DeriveStart
                    Exit Do
                  End If
                End If
                WorkFinal.MoveNext
              Loop
            End If
          End If
        Next

        ' check if there are operations that are not predecessor
        For i = 1 To ChildOprCount
          FinishOpr = 0
          FinishCount = 0
          WorkFinal.MoveFirst
          Do Until WorkFinal.EOF
            If ChildOpr(i) = WorkFinal("Predecessor") Then
              FinishOpr = -1
              FinishCount = FinishCount + 1
            End If
            WorkFinal.MoveNext
          Loop
          If FinishOpr <> -1 Then
            CheckFinish = ChildOpr(i)
            Call DeriveFinish
          ElseIf FinishOpr = -1 Then
            If FinishCount = 1 Then   ' is a predecessor but is cyclic
              WorkFinal.MoveFirst
              Do Until WorkFinal.EOF
                If ChildOpr(i) = WorkFinal("Predecessor") Then
                  If WorkFinal("Type") = 3 Then
                    CheckFinish = ChildOpr(i)
                    Call DeriveFinish
                    Exit Do
                  End If
                End If
                WorkFinal.MoveNext
              Loop
            End If
          End If
        Next
      End If
    End If
    End Sub

    Sub AnalyzeRelations (ChildOpr(), ChildOprCount, StatusOfChild)

      ' only one child operation case
    If ChildOprCount = 1 Then
      MyStart.AddNew
      MyStart("OprId") = TempOprId
      MyStart("Start") = ChildOpr(1)
      MyStart.Update
```

```
    MyFinish.AddNew
    MyFinish("OprId") = TempOprId
    MyFinish("Finish") = ChildOpr(1)
    MyFinish.Update
  Else
  ' clear the working database
    If WorkFinal.RecordCount > 0 Then WorkFinal.MoveFirst
    Do Until WorkFinal.EOF
      WorkFinal.Delete
      WorkFinal.MoveNext
    Loop

    ' collect all the operation sequencing for the current parent
    For i = 1 To ChildOprCount
      MyRel.MoveFirst
      Do Until MyRel.EOF
        If MyRel("Predecessor") = ChildOpr(i) Then
          WorkFinal.AddNew
          WorkFinal("Predecessor") = MyRel("Predecessor")
          WorkFinal("Successor") = MyRel("Successor")
          WorkFinal("Type") = MyRel("Type")
          WorkFinal("Parent") = MyRel("Parent")
          WorkFinal("ID") = MyRel("RelationId")
          WorkFinal.Update
        End If
        MyRel.MoveNext
      Loop
      MyRel.MoveFirst
      Do Until MyRel.EOF
        If MyRel("Successor") = ChildOpr(i) Then
          WorkFinal.AddNew
          WorkFinal("Predecessor") = MyRel("Predecessor")
          WorkFinal("Successor") = MyRel("Successor")
          WorkFinal("Type") = MyRel("Type")
          WorkFinal("Parent") = MyRel("Parent")
          WorkFinal("ID") = MyRel("RelationId")
          WorkFinal.Update
        End If•
        MyRel.MoveNext
      Loop
    Next

    'WorkFinal.MoveFirst
    'Printer.Print
    'Printer.Print
    'Do Until WorkFinal.EOF
    '   Printer.Print WorkFinal("ID"), WorkFinal("Predecessor"), WorkFinal("Successor")
    '   WorkFinal.MoveNext
    'Loop
    'Printer.EndDoc

    ' clear all duplicate relations
    If WorkFinal.RecordCount < 2 Then GoTo Try12
    WorkFinal.Index = "IDIX"
```

```
  WorkFinal.MoveFirst
  Do Until WorkFinal.EOF
    Temp1 = WorkFinal("ID")
Try11:
    If WorkFinal.RecordCount >= 2 Then
      WorkFinal.MoveNext
    Else
      Exit Do
    End If
    If WorkFinal("ID") = Temp1 Then
      WorkFinal.Delete
      WorkFinal.MoveNext
      If WorkFinal.EOF Then
        Exit Do
      Else
        WorkFinal.MovePrevious
      End If
      GoTo Try11:
    Else
      WorkFinal.Seek "=", Temp1
      WorkFinal.MoveNext
    End If
  Loop

Try12:
  ' check if there is only one relation
  WorkFinal.MoveFirst
  If WorkFinal.RecordCount = 1 Then
    If WorkFinal("Type") = 1 Then
      MyStart.AddNew
      MyStart("OprID") = TempOprId
      MyStart("Start") = WorkFinal("Predecessor")
      MyStart.Update
      MyFinish.AddNew
      MyFinish("OprId") = TempOprId
      MyFinish("Finish") = WorkFinal("Successor")
      MyFinish.Update
    ElseIf WorkFinal("Type") = 2 Or WorkFinal("Type") = 4 Then
      MyStart.AddNew
      MyStart("OprId") = TempOprId
      MyStart("Start") = WorkFinal("Predecessor")
      MyStart.Update
      MyStart.AddNew
      MyStart("OprId") = TempOprId
      MyStart("Start") = WorkFinal("Successor")
      MyStart.Update
      MyFinish.AddNew
      MyFinish("OprId") = TempOprId
      MyFinish("Finish") = WorkFinal("Predecessor")
      MyFinish.Update
      MyFinish.AddNew
      MyFinish("OprId") = TempOprId
      MyFinish("Finish") = WorkFinal("Successor")
      MyFinish.Update
```

```
    ElseIf WorkFinal("Type") = 3 Then
      ' there is a user mistake in the operation sequencing
    End If
  Else ' case when more than one relation
    ' first check all the hammock operations
    WorkFinal.MoveFirst
    Do Until WorkFinal.EOF
      If WorkFinal("Type") = 4 Then
        MyStart.AddNew
        MyStart("OprId") = TempOprId
        MyStart("Start") = WorkFinal("Predecessor")
        MyStart.Update
        MyFinish.AddNew
        MyFinish("OprId") = TempOprId
        MyFinish("Finish") = WorkFinal("Successor")
        MyFinish.Update
        WorkFinal.Delete
      End If
      WorkFinal.MoveNext
    Loop

    ' check if there are operations that are not successors to get the start
    For i = 1 To ChildOprCount
      StartOpr = 0
      StartCount = 0
      WorkFinal.MoveFirst
      Do Until WorkFinal.EOF
        If ChildOpr(i) = WorkFinal("Successor") Then
          StartOpr = -1
          StartCount = StartCount + 1
        End If
        WorkFinal.MoveNext
      Loop
      If StartOpr <> -1 Then
        MyStart.AddNew
        MyStart("OprId") = TempOprId
        MyStart("Start") = ChildOpr(i)
        MyStart.Update
      ElseIf StartOpr = -1 Then  ' is a successor but is cyclic
        If StartCount = 1 Then
          WorkFinal.MoveFirst
          Do Until WorkFinal.EOF
            If ChildOpr(i) = WorkFinal("Successor") Then
              If WorkFinal("Type") = 3 Then
                MyStart.AddNew
                MyStart("OprId") = TempOprId
                MyStart("Start") = ChildOpr(i)
                MyStart.Update
                Exit Do
              End If
            End If
            WorkFinal.MoveNext
          Loop
        End If
```

```
      End If
    Next

    ' check if there are operations that are not predecessor
    For i = 1 To ChildOprCount
      FinishOpr = 0
      FinishCount = 0
      WorkFinal.MoveFirst•
      Do Until WorkFinal.EOF
        If ChildOpr(i) = WorkFinal("Predecessor") Then
          FinishOpr = -1
          FinishCount = FinishCount + 1
        End If
        WorkFinal.MoveNext
      Loop
      If FinishOpr <> -1 Then
        MyFinish.AddNew
        MyFinish("OprId") = TempOprId
        MyFinish("Finish") = ChildOpr(i)
        MyFinish.Update
      ElseIf FinishOpr = -1 Then
        If FinishCount = 1 Then   ' is a predecessor but is cyclic
          WorkFinal.MoveFirst
          Do Until WorkFinal.EOF
            If ChildOpr(i) = WorkFinal("Predecessor") Then
              If WorkFinal("Type") = 3 Then
                MyFinish.AddNew
                MyFinish("OprId") = TempOprId
                MyFinish("Finish") = ChildOpr(i)
                MyFinish.Update•
                Exit Do
              End If
            End If
            WorkFinal.MoveNext
          Loop
        End If
      End If
    Next
  End If
End If
End Sub

Sub BackupFile ()

Msg$ = "This step would delete information from last translation." & Chr$(10) + Chr$(13)
Msg$ = Msg$ + "          Do you want to continue?"
Response = MsgBox(Msg$, 36)
If Response = 7 Then
  Exit Sub
End If

TranForm!Frame3D1.Visible = True
TranForm!Panel3D1.Visible = True
TranForm!Panel3D1.FloodPercent = 1
```

```
' first make a copy of the project database
FileCopy "blank.mdb", "temp007.mdb"
' database for the project
Set WorkDb = OpenDatabase("temp007.mdb", True, False)
Set WorkOpr = WorkDb.OpenTable("OprTable") ' operation table
Set WorkRel = WorkDb.OpenTable("RelationTable")  ' operation sequencing table
Set WorkProcess = WorkDb.OpenTable("ProTable") ' Process table
Set WorkLine = WorkDb.OpenTable("Position") ' Position of the Operations
Set WorkLine1 = WorkDb.OpenTable("Position1")  ' Position of the Process
Set WorkRes = WorkDb.OpenTable("ResourceTable") ' Resource for the project
TranForm!Panel3D1.FloodPercent = 2
Set WorkNodes = WorkDb.OpenTable("NodeTable")  ' Nodes for the process
Set WorkCon = WorkDb.OpenTable("ConnectionTable")  ' connection for process nodes
Set WorkProInter = WorkDb.OpenTable("FollowTable") ' Process inter-dependency table
Set WorkCrew = WorkDb.OpenTable("CrewTable")  ' crew combination table
Set WorkAllocate = WorkDb.OpenTable("AllocateTable") ' Allocation of the resoucres to
operation
Set WorkArn = WorkDb.OpenTable("ArnFrnTable") ' allocate and free resource nodes
Set WorkFinal = WorkDb.OpenTable("FinalRelation")  ' final relations for the project
Set WorkStart = WorkDb.OpenTable("StartTable")   ' determined starts
Set WorkFinish = WorkDb.OpenTable("FinishTable")   ' determined finish
TranForm!Panel3D1.FloodPercent = 3
' convert the operation sequencing information
Call LowestOprLink

End Sub

Sub DeriveFinish ()

   MyOpr.MoveFirst
   Do Until MyOpr.EOF
      If MyOpr("OprId") = CheckFinish Then
         ' if the only child operation is lowest
         If MyOpr("Lowest") = True And MyOpr("OprType") <> 1 Then
            MyFinish.AddNew
            MyFinish("OprId") = TempOprId
            MyFinish("Finish") = CheckFinish
            MyFinish.Update
            Exit Do
         Else
         ' if the child is not lowest then determine the start and finish
            If MyFinish.RecordCount > 0 Then MyFinish.MoveFirst
            Do Until MyFinish.EOF
               If MyFinish("OprId") = CheckFinish Then
                  WorkFinish.AddNew
                  WorkFinish("OprId") = MyFinish("OprId")
                  WorkFinish("Finish") = MyFinish("Finish")
                  WorkFinish.Update
               End If
               MyFinish.MoveNext
            Loop
            If WorkFinish.RecordCount > 0 Then WorkFinish.MoveFirst
            Do Until WorkFinish.EOF
```

```
            MyFinish.AddNew
            MyFinish("OprId") = TempOprId
            MyFinish("Finish") = WorkFinish("Finish")
            MyFinish.Update
            WorkFinish.MoveNext
          Loop
        End If
      End If
      MyOpr.MoveNext
    Loop

End Sub

Sub DeriveStart ()

    MyOpr.MoveFirst
    Do Until MyOpr.EOF
      If MyOpr("OprId") = CheckStart Then
        ' if the only child operation is lowest
        If MyOpr("Lowest") = True And MyOpr("OprType") <> 1 Then
          MyStart.AddNew
          MyStart("OprId") = TempOprId
          MyStart("Start") = CheckStart
          MyStart.Update
          Exit Do
        Else
        ' if the child is not lowest then determine the start and finish
          If MyStart.RecordCount > 0 Then MyStart.MoveFirst
          Do Until MyStart.EOF
            If MyStart("OprId") = CheckStart Then
              WorkStart.AddNew
              WorkStart("OprId") = MyStart("OprId")
              WorkStart("Start") = MyStart("Start")
              WorkStart.Update
            End If
            MyStart.MoveNext
          Loop
          If WorkStart.RecordCount > 0 Then WorkStart.MoveFirst
          Do Until WorkStart.EOF
            MyStart.AddNew
            MyStart("OprId") = TempOprId
            MyStart("Start") = WorkStart("Start")
            MyStart.Update
            WorkStart.MoveNext
          Loop•
        End If
      End If
      MyOpr.MoveNext
    Loop
End Sub

Sub DisOprRel ()

    If MyFinal.RecordCount > 0 Then MyFinal.MoveFirst
```

```
  Do Until MyFinal.EOF
     If MyFinal("Type") = 1 Then
        A$ = "Serial"
     ElseIf MyFinal("Type") = 2 Then
        A$ = "Parallel"
     ElseIf MyFinal("Type") = 3 Then
        A$ = "Cyclic"
     ElseIf MyFinal("Type") = 4 Then
        A$ = "Hammock"
     End If
     TempText$ = TempText$ + "Predecessor Operation: " + Str$(MyFinal("Predecessor")) + "
Successor Operation: " + Str$(MyFinal("Successor")) + " Parent ID: " + Str$(MyFinal("Parent")) +
" Relation Type: " + A$ + Chr$(13) + Chr$(10)
     MyFinal.MoveNext
  Loop
  TranForm!Text1.Text = TempText$
  TranForm!Text1.Visible = True
End Sub

Sub LowestOprLink ()

' clear all old translation if any from project
If MyFinal.RecordCount > 0 Then MyFinal.MoveFirst
Do Until MyFinal.EOF
  MyFinal.Delete
  MyFinal.MoveNext
Loop
If MyFinish.RecordCount > 0 Then MyFinish.MoveFirst
Do Until MyFinish.EOF
  MyFinish.Delete
  MyFinish.MoveNext
Loop
If MyStart.RecordCount > 0 Then MyStart.MoveFirst
Do Until MyStart.EOF
  MyStart.Delete
  MyStart.MoveNext
Loop

' copy all operations to the work database
MyOpr.MoveFirst
Do Until MyOpr.EOF
  MyOpr.Edit
  MyOpr("Level") = 1
  MyOpr.Update
  MyOpr.MoveNext
Loop

TranForm!Panel3D1.FloodPercent = 3

MyOpr.MoveFirst
Do Until MyOpr.EOF
  If MyOpr("ParentId") <> 0 Then
     WorkOpr.AddNew
     WorkOpr("OpName") = MyOpr("OpName")
```

```
      WorkOpr("OprId") = MyOpr("OprId")
      WorkOpr("Code") = MyOpr("Code")
      WorkOpr("ParentId") = MyOpr("ParentId")
      WorkOpr("Lowest") = MyOpr("Lowest")
      WorkOpr("OX") = MyOpr("OX")
      WorkOpr("OY") = MyOpr("OY")
      WorkOpr("OprType") = MyOpr("OprType")
      WorkOpr("ParentName") = MyOpr("ParentName")
      WorkOpr("Level") = 1
      WorkOpr.Update
   End If
   MyOpr.MoveNext
Loop

' assign levels to all the project operations
MyOpr.MoveFirst
Do Until MyOpr.EOF
   If MyOpr("ParentId") = 0 Then
      MyOpr.Edit
      MyOpr("Level") = 1
      MyOpr.Update
   End If
   MyOpr.MoveNext
Loop

TranForm!Panel3D1.FloodPercent = 4
WorkOpr.Index = "ParentIdIx"
MyOpr.Index = "OprIdIx"
WorkOpr.MoveFirst
Do Until WorkOpr.EOF
   MyOpr.Seek "=", WorkOpr("ParentId")
   WorkOpr.Edit
   WorkOpr("Level") = MyOpr("Level") + 1
   WorkOpr.Update
   MyOpr.Seek "=", WorkOpr("OprId")
   MyOpr.Edit
   MyOpr("Level") = WorkOpr("Level")
   MyOpr.Update
   WorkOpr.MoveNext
Loop

TranForm!Panel3D1.FloodPercent = 5

' collect all the lowest level operations
ReDim LowLevelOpr(50)
ReDim LowLevelParent(50)

LowOprCount = 0
MyOpr.Index = "ParentIdIx"
If MyOpr.RecordCount > 0 Then MyOpr.MoveFirst
Do Until MyOpr.EOF
   If MyOpr("Lowest") = True And MyOpr("OprType") <> 1 Then
      LowOprCount = LowOprCount + 1
      LowLevelOpr(LowOprCount) = MyOpr("OprId")
```

```
      LowLevelParent(LowOprCount) = MyOpr("ParentId")
    End If
    MyOpr.MoveNext
  Loop

  ' copy each lowest level relation from the relation table to the final table
  For i = 1 To LowOprCount
    TranForm!Panel3D1.FloodPercent = i * 10 / LowOprCount
    MyRel.MoveFirst
    Do Until MyRel.EOF
      If MyRel("Predecessor") = LowLevelOpr(i) Then
        TempSuc = MyRel("Successor")
        For n = 1 To LowOprCount
          If TempSuc = LowLevelOpr(n) Then
            WorkFinal.AddNew
            WorkFinal("Predecessor") = MyRel("Predecessor")
            WorkFinal("Successor") = MyRel("Successor")
            WorkFinal("Type") = MyRel("Type")
            WorkFinal("Parent") = MyRel("Parent")
            WorkFinal("ID") = MyRel("RelationId")
            WorkFinal.Update
            Exit For
          End If
        Next
      End If
      MyRel.MoveNext
    Loop
    MyRel.MoveFirst
    Do Until MyRel.EOF
      If MyRel("Successor") = LowLevelOpr(i) Then
        TempPre = MyRel("Predecessor")
        For n = 1 To LowOprCount
          If TempPre = LowLevelOpr(n) Then
            WorkFinal.AddNew
            WorkFinal("Predecessor") = MyRel("Predecessor")
            WorkFinal("Successor") = MyRel("Successor")
            WorkFinal("Type") = MyRel("Type")
            WorkFinal("Parent") = MyRel("Parent")
            WorkFinal("ID") = MyRel("RelationId")
            WorkFinal.Update
            Exit For
          End If
        Next
      End If
      MyRel.MoveNext
    Loop
  Next


  ' remove duplicates
  TranForm!Panel3D1.FloodPercent = 11
  WorkFinal.Index = "IDIX"
  WorkFinal.MoveFirst
  Do Until WorkFinal.EOF
```

```
    MyFinal.AddNew
    MyFinal("Predecessor") = WorkFinal("Predecessor")
    MyFinal("Successor") = WorkFinal("Successor")
    MyFinal("Type") = WorkFinal("Type")
    MyFinal("Parent") = WorkFinal("Parent")
    MyFinal("ID") = WorkFinal("ID")
    TempId = WorkFinal("ID")
    MyFinal.Update
    WorkFinal.Delete
    If WorkFinal.RecordCount > 0 Then WorkFinal.MoveFirst
    Do Until WorkFinal.EOF
       If WorkFinal("ID") = TempId Then
          WorkFinal.Delete
       End If
       WorkFinal.MoveNext
    Loop
    If WorkFinal.RecordCount > 0 Then WorkFinal.MoveFirst
Loop

TranForm!Panel3D1.FloodPercent = 12
t1 = 12
' based on the levels of the operation start analysis (not lowest)
MyOpr.Index = "LevelIdIx"
MyOpr.MoveFirst
Do Until MyOpr.EOF
   ChildOprCount = 0
   If MyOpr("Lowest") = False Or MyOpr("OprType") = 1 Then
      TempOprId = MyOpr("OprId")
      MyOpr.MoveFirst
      Do Until MyOpr.EOF
         If MyOpr("ParentId") = TempOprId Then
            ChildOprCount = ChildOprCount + 1
            ChildOpr(ChildOprCount) = MyOpr("OprId")
         End If
         MyOpr.MoveNext
      Loop
      ' subroutine to analyze the children
      Call AnalyzeChild(ChildOpr(), ChildOprCount, StatusOfChild)
      If StatusOfChild = 1 Then ' all are lowest level
         Call AnalyzeRelations(ChildOpr(), ChildOprCount, StatusOfChild)
      Else ' not lowest case ******
         Call AnalyzeNotLow(ChildOpr(), ChildOprCount)
      End If
      t1 = t1 + 1
      If t1 <= 50 Then
         TranForm!Panel3D1.FloodPercent = t1
      End If
      MyOpr.MoveFirst
      Do Until MyOpr.EOF
         If MyOpr("OprId") = TempOprId Then
            Exit Do
         End If
         MyOpr.MoveNext
      Loop
```

```
    End If
    MyOpr.MoveNext
  Loop

  TranForm!Panel3D1.FloodPercent = 51
  ' after determining the start & finish of the operations at all
  ' levels in terms of the lowest level operations derive the links
  If WorkFinal.RecordCount > 0 Then WorkFinal.MoveFirst
  Do Until WorkFinal.EOF
    WorkFinal.Delete
    WorkFinal.MoveNext
  Loop

  TranForm!Panel3D1.FloodPercent = 52
  If MyRel.RecordCount > 0 Then MyRel.MoveFirst
  Do Until MyRel.EOF
    MyFinal.Index = "IDIX"
    MyFinal.Seek "=", MyRel("RelationID")
    If MyFinal.NoMatch Then
      WorkFinal.AddNew
      WorkFinal("Predecessor") = MyRel("Predecessor")
      WorkFinal("Successor") = MyRel("Successor")
      WorkFinal("Type") = MyRel("Type")
      WorkFinal("Parent") = MyRel("Parent")
      WorkFinal("ID") = MyRel("RelationId")
      WorkFinal.Update
    End If
    MyRel.MoveNext
  Loop

  TranForm!Panel3D1.FloodPercent = 60
  ReDim DerivedPre(50), DerivedSuc(50)
  If WorkFinal.RecordCount > 0 Then WorkFinal.MoveFirst
  Do Until WorkFinal.EOF
    PreCount = 0
    SucCount = 0
    If WorkFinal("Type") = 1 Then
      MyFinish.MoveFirst
      Do Until MyFinish.EOF
        If MyFinish("OprID") = WorkFinal("Predecessor") Then
          PreCount = PreCount + 1
          DerivedPre(PreCount) = MyFinish("Finish")
        End If
        MyFinish.MoveNext
      Loop•
      If MyStart.RecordCount > 0 Then MyStart.MoveFirst
      Do Until MyStart.EOF
        If MyStart("OprID") = WorkFinal("Successor") Then
          SucCount = SucCount + 1
          DerivedSuc(SucCount) = MyStart("Start")
        End If
        MyStart.MoveNext
      Loop
      For i = 1 To PreCount
```

```
      For j = 1 To SucCount
        MyFinal.AddNew
        MyFinal("Predecessor") = DerivedPre(i)
        MyFinal("Successor") = DerivedSuc(j)
        MyFinal("Type") = 1
        MyFinal("Parent") = WorkFinal("Parent")
        MyFinal("ID") = WorkFinal("ID")
        MyFinal.Update
      Next
    Next
  ElseIf WorkFinal("Type") = 2 Then•
    MyFinish.MoveFirst
    Do Until MyFinish.EOF
      If MyFinish("OprID") = WorkFinal("Predecessor") Then
        PreCount = PreCount + 1
        DerivedPre(PreCount) = MyFinish("Finish")
      End If
      MyFinish.MoveNext
    Loop
    MyStart.MoveFirst
    Do Until MyStart.EOF
      If MyStart("OprID") = WorkFinal("Successor") Then
        SucCount = SucCount + 1
        DerivedSuc(SucCount) = MyStart("Start")
      End If
      MyFinish.MoveNext
    Loop
    For i = 1 To PreCount
      For j = 1 To SucCount
        MyFinal.AddNew
        MyFinal("Predecessor") = DerivedPre(i)
        MyFinal("Successor") = DerivedSuc(j)
        MyFinal("Type") = 2
        MyFinal("Parent") = WorkFinal("Parent")
        MyFinal("ID") = WorkFinal("ID")
        MyFinal.Update
      Next
    Next
  ElseIf WorkFinal("Type") = 3 Then
    MyFinish.MoveFirst
    Do Until MyFinish.EOF
      If MyFinish("OprID") = WorkFinal("Predecessor") Then•
        PreCount = PreCount + 1
        DerivedPre(PreCount) = MyFinish("Finish")
      End If
      MyFinish.MoveNext
    Loop
    MyStart.MoveFirst
    Do Until MyStart.EOF
      If MyStart("OprID") = WorkFinal("Successor") Then
        SucCount = SucCount + 1
        DerivedSuc(SucCount) = MyStart("Start")
      End If
      MyFinish.MoveNext
```

```
    Loop
    For i = 1 To PreCount
      For j = 1 To SucCount
        MyFinal.AddNew
        MyFinal("Predecessor") = DerivedPre(i)
        MyFinal("Successor") = DerivedSuc(j)
        MyFinal("Type") = 3
        MyFinal("Parent") = WorkFinal("Parent")
        MyFinal("ID") = WorkFinal("ID")
        MyFinal.Update
      Next
    Next
  ElseIf WorkFinal("Type") = 4 Then
    MyFinish.MoveFirst
    Do Until MyFinish.EOF
      If MyFinish("OprID") = WorkFinal("Predecessor") Then
        PreCount = PreCount + 1
        DerivedPre(PreCount) = MyFinish("Finish")
      End If
      MyFinish.MoveNext
    Loop
    MyStart.MoveFirst
    Do Until MyStart.EOF
      If MyStart("OprID") = WorkFinal("Predecessor") Then
        SucCount = SucCount + 1
        DerivedSuc(SucCount) = MyStart("Start")
      End If
      MyFinish.MoveNext
    Loop
    For i = 1 To PreCount
      For j = 1 To SucCount
        MyFinal.AddNew
        MyFinal("Predecessor") = DerivedPre(i)
        MyFinal("Successor") = DerivedSuc(j)
        MyFinal("Type") = 4
        MyFinal("Parent") = WorkFinal("Parent")
        MyFinal("ID") = WorkFinal("ID")
        MyFinal.Update
      Next
    Next
  End If
  WorkFinal.MoveNext
Loop

' after completing the operation sequening start resource listing
TranForm!Panel3D1.FloodPercent = 100
TranForm!Panel3D1.Visible = False
TranForm!Frame3D1.Visible = False
TranForm!Label1.Caption = "Translation Complete. Choose DISPLAY to see results."
TranForm!Label1.Visible = True
End Sub

Sub ResourceList ()
```

```
  If MyRes.RecordCount > 0 Then MyRes.MoveLast
  NumRes = MyRes.RecordCount
  ReDim ResText(NumRes)  As String

  MyRes.Index = "ResIDIx"
  If MyRes.RecordCount > 0 Then MyRes.MoveFirst
  CountRes = 1
  Do Until MyRes.EOF
     ResText(CountRes) = MyRes("ResourceName") & "(" & Str$(MyRes("ResourceID")) & ")/ "
     CountRes = CountRes + 1
     MyRes.MoveNext
  Loop

  For i = 1 To NumRes
     MyArn.Index = "PrId"
     If MyArn.RecordCount > 0 Then MyArn.MoveFirst
     Do Until MyArn.EOF
        If MyArn("ResourceId") = i Then
           ResText(i) = ResText(i) & Str$(MyArn("ProcessId")) & ","
        End If
        MyArn.MoveNext
     Loop
     MyArn.MoveFirst
  Next

  TranForm!Text1.Visible = True
  For i = 1 To NumRes
     TempText$ = TempText$ + ResText(i) + Chr$(13) + Chr$(10)
  Next
  TranForm!Text1.Text = TempText$
End Sub
```

Appendix C: Results of the case study

Table C-1: Processes for the Peace River bridge project

Process Number	Process
1	EXCAVTION PIER-1
2	PILLING ABT-1
3	PILING PIER-1
4	BLINDING LY PIER-1
5	FOOTNG FORM PIER-1
6	FOOTNG REBAR PIER-1
7	FOOTNG CONC PIER-1
8	SHAFT FORM PIER-1
9	SHAFT REBAR PIER-1
10	SHAFT CONC PIER-1
11	SHAFT FORM PIER-12
12	SHAFT REBAR PIER-12
13	SHAFT CONC PIER-12
14	CAP FORM PIER-1
15	CAP REBAR PIER-1
16	CAP CONC PIER-1
17	COFFERDAM PIER-2
18	BLINDING PIER-2
19	FOOTING FORM PIER-2
20	FOOTING REBAR PIER-2
21	FOOTING CONC PIER-2
22	PEDESTAL FORM PIER-2
23	PEDESTAL REBAR PIER-2
24	PILING ABT-2
25	PEDESTAL CONC PIER-2
26	PEDESTAL FORM PIER-22
27	PEDESTAL REBAR PIER-22
28	PEDESTAL CONC PIER-22
29	SHAFT FORM PIER-2
30	SHAFT REBAR PIER-2
31	SHAFT CONC PIER -2
32	SHAFT FORM PIER-22
33	SHAFT REBAR PIER-22
34	SHAFT CONC PIER-22
35	CAP FORM PIER-2
36	CAP REBAR PIER-2
37	CAP CONC PIER-2
38	BERM PIER-3
39	COFFER DAM PIER-3
40	BLINDING LY PIER-3

41	FOOTNG FORM PIER-3
42	FOOTNG REBAR PIER-3
43	FOOTNG CONC PIER-3
44	PEDESTAL FORM PIER-3
45	PEDESTAL REBAR PIER-3
46	PEDESTAL CONC PIER-3
47	PEDESTAL FORM PIER-32
48	PEDESTAL REBAR PIER-32
49	PEDESTAL CONC PIER-32
50	SHAFT FORM PIER-3
51	SHAFT REBAR PIER-3
52	SHAFT CONC PIER-3
53	SHAFT FORM PIER-32
54	SHAFT REBAR PIER-32
55	SHAFT CONC PIER-32
56	CAP FORM PIER-3
57	CAP REBAR PIER-3
58	CAP CONC PIER-3
59	BERM PIER-4
60	COFFERDAM PIER-4
61	BLINDNG LY PIER-4
62	FOOTNG FORM PIER-4
63	FOOTING REBAR PIER-4
64	FOOTNG CONC PIER-4
65	PEDESTAL FORM PIER-4
66	PEDESTAL REBAR PIER-4
67	PEDESTAL CONC.. PIER-4
68	PEDESTAL FORM PIER-42
69	PEDESTAL REBAR PIER-42
70	PEDESTAL CONC PIER-42
71	SHAFT FORM PIER-4
72	SHAFT REBAR PIER-4
73	SHAFT CONC PIER-4
74	SHAFT FORM PIER-42
75	SHAFT REBAR PIER-42
76	SHAFT CONC PIER-42
78	CAP FORM PIER-4
79	CAP REBAR PIER-4
80	CAP CONC PIER-4
81	BERM PIER-5
82	COFFERDAM PIER-5
83	BLINDING LY PIER-5
84	FOOTNG FORM PIER-5
85	FOOTNG REBAR PIER-5
86	FOOTING CONC PIER-5
87	PEDESTAL FORM PIER-5

89	PEDESTAL REBAR PIER-5
90	PEDESTAL CONC PIER-5
91	PEDESTAL FORM PIER-52
92	PEDESTAL REBAR PIER-52
93	PEDESTAL CONC PIER-52
94	SHAFT FORM PIER-5
95	SHAFT REBAR PIER-5
96	SHAFT CONC PIER-5
97	SHAFT FORM PIER-52
98	SHAFT REBAR PIER-52
99	SHAFT CONC PIER-52
100	CAP FORM PIER-5
101	CAP REBAR PIER-5
102	CAP CONC PIER-5
103	BLINDING LY PIER-6
104	FOOTNG FORM PIER-6
105	FOOTNG REBAR PIER-6
106	FOOTNG CONC PIER-6
107	PEDESTAL FORM PIER-6
108	PEDESTAL REBAR PIER-6
109	PEDESTAL CONC PIER-6
110	SHAFT FORM PIER-6
111	SHAFT REBAR PIER-6
112	SHAFT CONC PIER-6
113	SHAFT FORM PIER-62
114	SHAFT REBAR PIER-62
115	SHAFT CONC PIER-62
116	CAP FORM PIER-6
117	CAP REBAR PIER-6
118	CAP CONC PIER-6

Table C-2: Simulation results for the deterministic case

Process	Start (in days)	Finish (in days)	Duration (in days)
EXCAVTION PIER-1	0.00	15.35	15.35
PILLING ABT-1	0.00	6.00	6.00
PILING PIER-1	15.35	32.23	16.88
BLINDING LY PIER-1	32.23	32.36	0.14
FOOTNG FORM PIER-1	32.36	35.36	3.00
FOOTNG REBAR PIER-1	35.36	41.36	6.00
FOOTNG CONC PIER-1	41.36	44.20	2.84
SHAFT FORM PIER-1	43.20	46.20	3.00
SHAFT REBAR PIER-1	46.20	52.20	6.00
SHAFT CONC PIER-1	52.20	53.07	0.86
SHAFT FORM PIER-12	54.07	57.07	3.00
SHAFT REBAR PIER-12	57.07	63.07	6.00
SHAFT CONC PIER-12	63.07	63.93	0.86
CAP FORM PIER-1	63.93	65.93	2.00
CAP REBAR PIER-1	65.93	70.93	5.00
CAP CONC PIER-1	70.93	72.21	1.28
COFFERDAM PIER-2	72.21	131.29	59.07
BLINDING PIER-2	131.29	131.42	0.14
FOOTING FORM PIER-2	139.51	142.51	3.00
FOOTING REBAR PIER-2	148.51	154.51	6.00
FOOTING CONC PIER-2	154.51	157.52	3.01
PEDESTAL FORM PIER-2	157.83	162.83	5.00
PEDESTAL REBAR PIER-2	164.83	171.83	7.00
PILING ABT-2	167.66	167.85	0.19
PEDESTAL CONC PIER-2	171.83	173.21	1.39
PEDESTAL FORM PIER-22	174.92	179.92	5.00
PEDESTAL REBAR PIER-22	181.92	188.92	7.00
PEDESTAL CONC PIER-22	188.92	190.31	1.39
SHAFT FORM PIER-2	190.31	194.31	4.00
SHAFT REBAR PIER-2	194.92	201.92	7.00
SHAFT CONC PIER -2	201.92	203.02	1.09
SHAFT FORM PIER-22	204.02	208.02	4.00
SHAFT REBAR PIER-22	208.02	215.02	7.00
SHAFT CONC PIER-22	215.02	216.11	1.09
CAP FORM PIER-2	216.11	219.11	3.00
CAP REBAR PIER-2	219.11	225.11	6.00
CAP CONC PIER-2	225.11	226.54	1.43
BERM PIER-3	15.35	48.75	33.40
COFFER DAM PIER-3	48.75	99.04	50.29
BLINDING LY PIER-3	99.04	99.17	0.14
FOOTNG FORM PIER-3	99.17	102.17	3.00
FOOTNG REBAR PIER-3	102.17	108.17	6.00
FOOTNG CONC PIER-3	108.17	111.18	3.01
PEDESTAL FORM PIER-3	110.18	119.18	9.00

PEDESTAL REBAR PIER-3	119.18	128.18	9.00
PEDESTAL CONC PIER-3	128.18	129.51	1.32
PEDESTAL FORM PIER-32	130.51	139.51	9.00
PEDESTAL REBAR PIER-32	139.51	148.51	9.00
PEDESTAL CONC PIER-32	148.51	149.83	1.32
SHAFT FORM PIER-3	149.83	157.83	8.00
SHAFT REBAR PIER-3	157.83	164.83	7.00
SHAFT CONC PIER-3	164.83	165.92	1.09
SHAFT FORM PIER-32	166.92	174.92	8.00
SHAFT REBAR PIER-32	174.92	181.92	7.00
SHAFT CONC PIER-32	181.92	183.02	1.09
CAP FORM PIER-3	183.02	186.02	3.00
CAP REBAR PIER-3	188.92	194.92	6.00
CAP CONC PIER-3	194.92	196.35	1.43
BERM PIER-4	33.40	66.79	33.40
COFFERDAM PIER-4	66.79	163.54	96.75
BLINDNG LY PIER-4	163.54	163.67	0.14
FOOTNG FORM PIER-4	163.67	166.67	3.00
FOOTING REBAR PIER-4	166.67	172.67	6.00
FOOTNG CONC PIER-4	172.67	175.52	2.84
PEDESTAL FORM PIER-4	174.52	183.52	9.00
PEDESTAL REBAR PIER-4	183.52	192.52	9.00
PEDESTAL CONC.. PIER-4	192.52	193.99	1.47
PEDESTAL FORM PIER-42	194.99	203.99	9.00
PEDESTAL REBAR PIER-42	203.99	212.99	9.00
PEDESTAL CONC PIER-42	212.99	214.45	1.47
SHAFT FORM PIER-4	214.45	222.45	8.00
SHAFT REBAR PIER-4	222.45	229.45	7.00
SHAFT CONC PIER-4	229.45	230.55	1.09
SHAFT FORM PIER-42	231.55	239.55	8.00
SHAFT REBAR PIER-42	239.55	246.55	7.00
SHAFT CONC PIER-42	246.55	247.64	1.09
CAP FORM PIER-4	247.64	250.64	3.00
CAP REBAR PIER-4	250.64	256.64	6.00
CAP CONC PIER-4	256.64	257.80	1.15
BERM PIER-5	0.00	33.40	33.40
COFFERDAM PIER-5	33.54	66.79	33.25
BLINDING LY PIER-5	66.79	66.92	0.14
FOOTNG FORM PIER-5	66.92	69.92	3.00
FOOTNG REBAR PIER-5	69.92	75.92	6.00
FOOTING CONC PIER-5	75.92	78.77	2.84
PEDESTAL FORM PIER-5	77.77	80.77	3.00
PEDESTAL REBAR PIER-5	80.77	86.77	6.00
PEDESTAL CONC PIER-5	86.77	88.24	1.47
PEDESTAL FORM PIER-52	89.24	92.24	3.00
PEDESTAL REBAR PIER-52	92.24	98.24	6.00

PEDESTAL CONC PIER-52	98.24	99.70	1.47
SHAFT FORM PIER-5	99.70	105.70	6.00
SHAFT REBAR PIER-5	105.70	112.70	7.00
SHAFT CONC PIER-5	112.70	113.80	1.09
SHAFT FORM PIER-52	114.80	120.80	6.00
SHAFT REBAR PIER-52	120.80	127.80	7.00
SHAFT CONC PIER-52	127.80	128.89	1.09
CAP FORM PIER-5	128.89	131.89	3.00
CAP REBAR PIER-5	131.89	137.89	6.00
CAP CONC PIER-5	137.89	139.32	1.43
BLINDING LY PIER-6	0.00	0.14	0.14
FOOTNG FORM PIER-6	0.14	3.14	3.00
FOOTNG REBAR PIER-6	3.14	9.14	6.00
FOOTNG CONC PIER-6	9.14	11.98	2.84
PEDESTAL FORM PIER-6	10.98	15.98	5.00
PEDESTAL REBAR PIER-6	15.98	22.98	7.00
PEDESTAL CONC PIER-6	22.98	25.32	2.34
SHAFT FORM PIER-6	24.32	31.32	7.00
SHAFT REBAR PIER-6	31.32	38.32	7.00
SHAFT CONC PIER-6	38.32	39.35	1.03
SHAFT FORM PIER-62	40.35	47.35	7.00
SHAFT REBAR PIER-62	47.35	54.35	7.00
SHAFT CONC PIER-62	54.35	55.39	1.03
CAP FORM PIER-6	55.39	58.39	3.00
CAP REBAR PIER-6	58.39	64.39	6.00
CAP CONC PIER-6	64.39	65.77	1.39

Table C-3: Simulation results for the stochastic case (mean of 30 runs)

Process	Start (in days)	End (in days)	Duration (days)
EXCAVTION PIER-1	0.00	24.63	24.63
PILLING ABT-1	0.00	6.38	6.38
PILING PIER-1	24.63	42.51	17.88
BLINDING LY PIER-1	42.51	42.66	0.15
FOOTNG FORM PIER-1	42.66	45.66	3.00
FOOTNG REBAR PIER-1	45.66	51.56	5.90
FOOTNG CONC PIER-1	51.56	54.44	2.88
SHAFT FORM PIER-1	53.44	56.50	3.06
SHAFT REBAR PIER-1	56.50	62.48	5.98
SHAFT CONC PIER-1	62.48	63.39	0.91
SHAFT FORM PIER-12	64.39	67.36	2.97
SHAFT REBAR PIER-12	117.27	123.13	5.86
SHAFT CONC PIER-12	123.13	124.02	0.89
CAP FORM PIER-1	124.02	126.03	2.01
CAP REBAR PIER-1	129.16	134.22	5.07
CAP CONC PIER-1	134.22	135.53	1.31
COFFERDAM PIER-2	135.53	184.71	49.18
BLINDING PIER-2	184.71	184.87	0.16
FOOTING FORM PIER-2	184.87	187.91	3.04
FOOTING REBAR PIER-2	193.72	199.73	6.01
PILING ABT-2	195.07	195.29	0.22
FOOTING CONC PIER-2	199.73	202.82	3.09
PEDESTAL FORM PIER-2	223.38	228.41	5.03
PEDESTAL REBAR PIER-2	230.35	237.34	7.00
PEDESTAL CONC PIER-2	237.34	238.79	1.45
PEDESTAL FORM PIER-22	240.56	245.57	5.01
PEDESTAL REBAR PIER-22	247.55	254.58	7.04
PEDESTAL CONC PIER-22	254.58	256.00	1.42
SHAFT FORM PIER-2	256.00	260.10	4.10
SHAFT REBAR PIER-2	260.57	267.59	7.02
SHAFT CONC PIER -2	267.59	268.72	1.13
SHAFT FORM PIER-22	269.72	273.71	3.99
SHAFT REBAR PIER-22	273.71	280.70	6.99
SHAFT CONC PIER-22	280.70	281.81	1.11
CAP FORM PIER-2	281.81	284.79	2.97
CAP REBAR PIER-2	284.79	290.76	5.98
CAP CONC PIER-2	290.76	292.19	1.43
BERM PIER-3	24.63	58.31	33.67
COFFER DAM PIER-3	62.48	117.27	54.79
BLINDING LY PIER-3	117.27	117.41	0.14
FOOTNG FORM PIER-3	117.41	120.46	3.05
FOOTNG REBAR PIER-3	123.13	129.16	6.03
FOOTNG CONC PIER-3	129.16	132.21	3.05
PEDESTAL FORM PIER-3	131.21	140.20	8.99
PEDESTAL REBAR PIER-3	184.71	193.72	9.01

PEDESTAL CONC PIER-3	193.72	195.06	1.35
PEDESTAL FORM PIER-32	196.06	205.06	8.99
PEDESTAL REBAR PIER-32	205.06	214.05	8.99
PEDESTAL CONC PIER-32	214.05	215.42	1.38
SHAFT FORM PIER-3	215.42	223.38	7.96
SHAFT REBAR PIER-3	223.38	230.35	6.97
SHAFT CONC PIER-3	230.35	231.49	1.15
SHAFT FORM PIER-32	232.49	240.56	8.06
SHAFT REBAR PIER-32	240.56	247.55	6.99
SHAFT CONC PIER-32	247.55	248.68	1.13
CAP FORM PIER-3	248.68	251.70	3.03
CAP REBAR PIER-3	254.58	260.57	5.98
CAP CONC PIER-3	260.57	262.01	1.44
BERM PIER-4	30.29	60.49	30.19
COFFERDAM PIER-4	87.60	147.47	59.87
BLINDNG LY PIER-4	147.47	147.62	0.14
FOOTNG FORM PIER-4	167.38	170.37	2.99
FOOTING REBAR PIER-4	175.65	182.73	7.08
FOOTNG CONC PIER-4	182.73	185.61	2.88
PEDESTAL FORM PIER-4	198.39	208.51	10.13
PEDESTAL REBAR PIER-4	208.51	216.56	8.05
PEDESTAL CONC.. PIER-4	216.56	218.09	1.53
PEDESTAL FORM PIER-42	219.09	229.69	10.60
PEDESTAL REBAR PIER-42	229.69	232.92	3.23
PEDESTAL CONC PIER-42	232.92	234.43	1.51
SHAFT FORM PIER-4	234.43	242.23	7.80
SHAFT REBAR PIER-4	242.23	250.31	8.09
SHAFT CONC PIER-4	250.31	251.44	1.12
SHAFT FORM PIER-42	252.44	262.70	10.27
SHAFT REBAR PIER-42	262.70	272.32	9.62
SHAFT CONC PIER-42	272.32	273.44	1.12
CAP FORM PIER-4	273.44	277.46	4.02
CAP REBAR PIER-4	277.46	282.18	4.72
CAP CONC PIER-4	282.18	283.61	1.43
BERM PIER-5	0.00	30.29	30.29
COFFERDAM PIER-5	30.29	80.22	49.92
BLINDING LY PIER-5	80.22	80.37	0.15
FOOTNG FORM PIER-5	80.37	83.42	3.06
FOOTNG REBAR PIER-5	155.19	161.31	6.12
FOOTING CONC PIER-5	161.31	164.26	2.94
PEDESTAL FORM PIER-5	163.26	167.38	4.12
PEDESTAL REBAR PIER-5	167.38	175.65	8.28
PEDESTAL CONC PIER-5	175.65	177.16	1.51
PEDESTAL FORM PIER-52	178.16	182.27	4.11
PEDESTAL REBAR PIER-52	182.73	189.27	6.54
PEDESTAL CONC PIER-52	189.27	190.82	1.54
SHAFT FORM PIER-5	190.82	198.39	7.57

SHAFT REBAR PIER-5	198.39	207.74	9.36
SHAFT CONC PIER-5	207.74	208.86	1.12
SHAFT FORM PIER-52	209.86	215.44	5.58
SHAFT REBAR PIER-52	216.56	225.98	9.41
SHAFT CONC PIER-52	225.98	227.11	1.14
CAP FORM PIER-5	229.69	232.70	3.02
CAP REBAR PIER-5	232.92	239.44	6.53
CAP CONC PIER-5	239.44	240.90	1.46
BLINDING LY PIER-6	0.00	0.14	0.14
FOOTNG FORM PIER-6	0.14	3.15	3.01
FOOTNG REBAR PIER-6	3.15	9.17	6.02
FOOTNG CONC PIER-6	9.17	12.08	2.91
PEDESTAL FORM PIER-6	11.08	16.08	4.99
PEDESTAL REBAR PIER-6	16.08	23.00	6.92
PEDESTAL CONC PIER-6	23.00	25.41	2.42
SHAFT FORM PIER-6	24.41	31.43	7.01
SHAFT REBAR PIER-6	80.22	87.60	7.38
SHAFT CONC PIER-6	87.60	88.68	1.08
SHAFT FORM PIER-62	89.68	96.67	6.98
SHAFT REBAR PIER-62	147.47	155.19	7.72
SHAFT CONC PIER-62	155.19	156.25	1.05
CAP FORM PIER-6	156.25	159.20	2.95
CAP REBAR PIER-6	161.31	167.27	5.96
CAP CONC PIER-6	167.27	168.67	1.40

Table C-4: Simulation results for the resource scenario-1

Process	Start (in days)	End (in days)	Duration (days)
EXCAVTION PIER-1	0.00	24.63	24.63
PILLING ABT-1	0.00	6.38	6.38
PILING PIER-1	24.63	42.51	17.88
BLINDING LY PIER-1	42.51	42.66	0.15
FOOTNG FORM PIER-1	42.66	45.66	3.00
FOOTNG REBAR PIER-1	45.66	51.56	5.90
FOOTNG CONC PIER-1	51.56	54.44	2.88
SHAFT FORM PIER-1	53.44	56.50	3.06
SHAFT REBAR PIER-1	56.50	62.48	5.98
SHAFT CONC PIER-1	62.48	63.39	0.91
SHAFT FORM PIER-12	64.39	67.36	2.97
SHAFT REBAR PIER-12	117.27	123.13	5.86
SHAFT CONC PIER-12	123.13	124.02	0.89
CAP FORM PIER-1	124.02	126.03	2.01
CAP REBAR PIER-1	129.16	134.22	5.07
CAP CONC PIER-1	134.22	135.53	1.31
COFFERDAM PIER-2	135.53	184.71	49.18
BLINDING PIER-2	184.71	184.87	0.16
FOOTING FORM PIER-2	184.87	187.91	3.04
FOOTING REBAR PIER-2	193.72	199.73	6.01
PILING ABT-2	195.07	195.29	0.22
FOOTING CONC PIER-2	199.73	202.82	3.09
PEDESTAL FORM PIER-2	223.38	228.41	5.03
PEDESTAL REBAR PIER-2	230.35	237.34	7.00
PEDESTAL CONC PIER-2	237.34	238.79	1.45
PEDESTAL FORM PIER-22	240.56	245.57	5.01
PEDESTAL REBAR PIER-22	247.55	254.58	7.04
PEDESTAL CONC PIER-22	254.58	256.00	1.42
SHAFT FORM PIER-2	256.00	260.10	4.10
SHAFT REBAR PIER-2	260.57	267.59	7.02
SHAFT CONC PIER -2	267.59	268.72	1.13
SHAFT FORM PIER-22	269.72	273.71	3.99
SHAFT REBAR PIER-22	273.71	280.70	6.99
SHAFT CONC PIER-22	280.70	281.81	1.11
CAP FORM PIER-2	281.81	284.79	2.97
CAP REBAR PIER-2	284.79	290.76	5.98
CAP CONC PIER-2	290.76	294.70	3.94
BERM PIER-3	24.63	58.31	33.67
COFFER DAM PIER-3	62.48	117.27	54.79
BLINDING LY PIER-3	117.27	117.41	0.14
FOOTNG FORM PIER-3	117.41	120.46	3.05
FOOTNG REBAR PIER-3	123.13	129.16	6.03
FOOTNG CONC PIER-3	129.16	132.21	3.05

PEDESTAL FORM PIER-3	131.21	140.20	8.99
PEDESTAL REBAR PIER-3	184.71	193.72	9.01
PEDESTAL CONC PIER-3	193.72	195.06	1.35
PEDESTAL FORM PIER-32	196.06	205.06	8.99
PEDESTAL REBAR PIER-32	205.06	214.05	8.99
PEDESTAL CONC PIER-32	214.05	215.42	1.38
SHAFT FORM PIER-3	215.42	223.38	7.96
SHAFT REBAR PIER-3	223.38	230.35	6.97
SHAFT CONC PIER-3	230.35	231.49	1.15
SHAFT FORM PIER-32	232.49	240.56	8.06
SHAFT REBAR PIER-32	240.56	247.55	6.99
SHAFT CONC PIER-32	247.55	248.68	1.13
CAP FORM PIER-3	248.68	251.70	3.03
CAP REBAR PIER-3	254.58	260.57	5.98
CAP CONC PIER-3	260.57	262.01	1.44
BERM PIER-4	30.29	60.49	30.19
COFFERDAM PIER-4	87.60	147.47	59.87
BLINDNG LY PIER-4	147.47	147.62	0.14
FOOTNG FORM PIER-4	167.38	170.37	2.99
FOOTING REBAR PIER-4	175.65	182.73	7.08
FOOTNG CONC PIER-4	182.73	185.61	2.88
PEDESTAL FORM PIER-4	198.39	208.51	10.13
PEDESTAL REBAR PIER-4	208.51	216.56	8.05
PEDESTAL CONC.. PIER-4	216.56	218.09	1.53
PEDESTAL FORM PIER-42	219.09	229.69	10.60
PEDESTAL REBAR PIER-42	229.69	232.92	3.23
PEDESTAL CONC PIER-42	232.92	234.43	1.51
SHAFT FORM PIER-4	234.43	242.23	7.80
SHAFT REBAR PIER-4	242.23	250.31	8.09
SHAFT CONC PIER-4	250.31	251.44	1.12
SHAFT FORM PIER-42	252.44	262.70	10.27
SHAFT REBAR PIER-42	262.70	272.32	9.62
SHAFT CONC PIER-42	272.32	273.44	1.12
CAP FORM PIER-4	273.44	277.46	4.02
CAP REBAR PIER-4	277.46	282.18	4.72
CAP CONC PIER-4	282.18	283.61	1.43
BERM PIER-5	0.00	30.29	30.29
COFFERDAM PIER-5	30.29	80.22	49.92
BLINDING LY PIER-5	80.22	80.37	0.15
FOOTNG FORM PIER-5	80.37	83.42	3.06
FOOTNG REBAR PIER-5	155.19	161.31	6.12
FOOTING CONC PIER-5	161.31	164.26	2.94
PEDESTAL FORM PIER-5	163.26	167.38	4.12
PEDESTAL REBAR PIER-5	167.38	175.65	8.28
PEDESTAL CONC PIER-5	175.65	177.16	1.51
PEDESTAL FORM PIER-52	178.16	182.27	4.11
PEDESTAL REBAR PIER-52	182.73	189.27	6.54

PEDESTAL CONC PIER-52	189.27	190.82	1.54
SHAFT FORM PIER-5	190.82	198.39	7.57
SHAFT REBAR PIER-5	198.39	207.74	9.36
SHAFT CONC PIER-5	207.74	208.86	1.12
SHAFT FORM PIER-52	209.86	215.44	5.58
SHAFT REBAR PIER-52	216.56	225.98	9.41
SHAFT CONC PIER-52	225.98	227.11	1.14
CAP FORM PIER-5	229.69	232.70	3.02
CAP REBAR PIER-5	232.92	239.44	6.53
CAP CONC PIER-5	239.44	240.90	1.46
BLINDING LY PIER-6	0.00	0.14	0.14
FOOTNG FORM PIER-6	0.14	3.15	3.01
FOOTNG REBAR PIER-6	3.15	9.17	6.02
FOOTNG CONC PIER-6	9.17	12.08	2.91
PEDESTAL FORM PIER-6	11.08	16.08	4.99
PEDESTAL REBAR PIER-6	16.08	23.00	6.92
PEDESTAL CONC PIER-6	23.00	25.41	2.42
SHAFT FORM PIER-6	24.41	31.43	7.01
SHAFT REBAR PIER-6	80.22	87.60	7.38
SHAFT CONC PIER-6	87.60	88.68	1.08
SHAFT FORM PIER-62	89.68	96.67	6.98
SHAFT REBAR PIER-62	147.47	155.19	7.72
SHAFT CONC PIER-62	155.19	156.25	1.05
CAP FORM PIER-6	156.25	159.20	2.95
CAP REBAR PIER-6	161.31	167.27	5.96
CAP CONC PIER-6	167.27	168.67	1.40

Table C-5: Simulation results for the revised operation sequening case

PROCESS NAME	Start (in days)	Finish (in days)	Duration (in days)
BLINDING LY PIER-6	0.00	0.15	0.15
FOOTNG FORM PIER-6	0.15	3.17	3.02
PILLING ABT-1	0.00	6.41	6.41
FOOTNG REBAR PIER-6	3.17	9.15	5.98
FOOTNG CONC PIER-6	9.15	12.07	2.93
EXCAVTION PIER-1	0.00	15.00	15.00
PEDESTAL FORM PIER-6	11.07	16.05	4.98
PEDESTAL REBAR PIER-6	16.05	23.03	6.98
PEDESTAL CONC PIER-6	23.03	25.41	2.38
BERM PIER-5	0.00	30.63	30.63
SHAFT FORM PIER-6	24.41	31.40	6.99
BERM PIER-3	0.00	33.99	33.99
SHAFT REBAR PIER-6	31.40	38.78	7.38
SHAFT CONC PIER-6	38.78	39.81	1.03
COFFERDAM PIER-2	0.00	43.49	43.49
BLINDING PIER-2	43.49	43.64	0.16
FOOTING FORM PIER-2	43.64	46.64	3.00
SHAFT FORM PIER-62	40.81	47.76	6.95
FOOTING REBAR PIER-2	46.64	52.60	5.96
SHAFT REBAR PIER-62	47.76	54.75	6.99
FOOTING CONC PIER-2	52.60	55.66	3.05
SHAFT CONC PIER-62	54.75	55.79	1.03
CAP FORM PIER-6	55.79	58.75	2.96
PEDESTAL FORM PIER-2	54.66	59.66	5.01
BERM PIER-4	30.63	60.78	30.15
PILING PIER-1	15.00	61.60	46.59
BLINDING LY PIER-1	61.60	61.75	0.15
CAP REBAR PIER-6	58.75	64.72	5.97
FOOTNG FORM PIER-1	61.75	64.72	2.97
CAP CONC PIER-6	64.72	66.12	1.41
PEDESTAL REBAR PIER-2	59.66	66.66	7.00
PEDESTAL CONC PIER-2	66.66	68.10	1.44
FOOTNG REBAR PIER-1	66.66	72.69	6.02
PEDESTAL FORM PIER-22	69.10	74.18	5.08
FOOTNG CONC PIER-1	72.69	75.60	2.91
SHAFT FORM PIER-1	74.60	77.70	3.10
PEDESTAL REBAR PIER-22	74.18	81.18	7.00
PEDESTAL CONC PIER-22	81.18	82.60	1.42
SHAFT REBAR PIER-1	81.18	87.17	6.00
SHAFT CONC PIER-1	87.17	88.04	0.87
SHAFT FORM PIER-12	89.04	92.10	3.06
SHAFT REBAR PIER-12	92.10	97.97	5.87
COFFER DAM PIER-3	33.99	98.31	64.32

SHAFT CONC PIER-12	97.97	98.88	0.91
BLINDING LY PIER-3	98.31	98.99	0.69
CAP FORM PIER-1	98.88	100.90	2.01
SHAFT FORM PIER-2	100.90	104.87	3.97
CAP REBAR PIER-1	100.90	105.41	4.51
CAP CONC PIER-1	105.41	106.69	1.28
FOOTNG FORM PIER-3	104.87	107.91	3.04
SHAFT REBAR PIER-2	105.41	112.42	7.01
SHAFT CONC PIER -2	112.42	113.55	1.13
FOOTNG REBAR PIER-3	112.42	118.34	5.92
SHAFT FORM PIER-22	114.55	118.50	3.95
FOOTNG CONC PIER-3	118.34	121.40	3.06
SHAFT REBAR PIER-22	118.50	125.45	6.95
SHAFT CONC PIER-22	125.45	126.58	1.13
PEDESTAL FORM PIER-3	120.40	129.41	9.02
CAP FORM PIER-2	129.41	132.38	2.97
COFFERDAM PIER-5	30.63	136.16	105.52
BLINDING LY PIER-5	136.16	136.30	0.15
PEDESTAL REBAR PIER-3	129.41	138.41	9.00
FOOTNG FORM PIER-5	136.30	139.42	3.12
PEDESTAL CONC PIER-3	138.41	139.79	1.38
CAP REBAR PIER-2	138.41	144.38	5.97
FOOTNG REBAR PIER-5	139.42	145.44	6.01
CAP CONC PIER-2	144.38	145.84	1.46
FOOTING CONC PIER-5	145.44	148.34	2.90
PEDESTAL FORM PIER-32	140.79	149.79	9.00
PEDESTAL FORM PIER-5	147.34	151.26	3.92
PEDESTAL REBAR PIER-32	149.79	158.77	8.97
PEDESTAL CONC PIER-32	158.77	160.15	1.38
PEDESTAL REBAR PIER-5	151.26	161.08	9.83
PEDESTAL CONC PIER-5	161.08	162.64	1.56
COFFERDAM PIER-4	64.72	166.15	101.43
BLINDNG LY PIER-4	166.15	166.31	0.16
PEDESTAL FORM PIER-52	163.64	167.64	3.99
SHAFT FORM PIER-3	160.15	168.16	8.01
FOOTNG FORM PIER-4	167.64	170.57	2.93
PILING ABT-2	171.93	172.11	0.18
SHAFT REBAR PIER-3	168.16	175.19	7.03
PEDESTAL REBAR PIER-52	167.64	175.84	8.21
SHAFT CONC PIER-3	175.19	176.35	1.16
PEDESTAL CONC PIER-52	175.84	177.36	1.51
FOOTING REBAR PIER-4	175.84	181.49	5.64
FOOTNG CONC PIER-4	181.49	184.37	2.88
SHAFT FORM PIER-32	177.35	185.28	7.93

SHAFT FORM PIER-5	177.36	185.34	7.99
SHAFT REBAR PIER-5	185.34	191.23	5.89
SHAFT REBAR PIER-32	185.28	192.30	7.02
SHAFT CONC PIER-5	191.23	192.35	1.11
SHAFT CONC PIER-32	192.30	193.43	1.13
PEDESTAL FORM PIER-4	185.34	194.72	9.38
CAP FORM PIER-3	193.43	196.41	2.99
SHAFT FORM PIER-52	194.72	202.01	7.29
CAP REBAR PIER-3	196.41	202.42	6.00
CAP CONC PIER-3	202.42	203.84	1.42
PEDESTAL REBAR PIER-4	194.72	206.04	11.32
PEDESTAL CONC.. PIER-4	206.04	207.55	1.51
SHAFT REBAR PIER-52	206.04	212.05	6.01
SHAFT CONC PIER-52	212.05	213.18	1.13
PEDESTAL FORM PIER-42	208.55	216.43	7.88
CAP FORM PIER-5	216.43	219.25	2.82
PEDESTAL REBAR PIER-42	216.43	220.81	4.37
PEDESTAL CONC PIER-42	220.81	222.33	1.53
CAP REBAR PIER-5	220.81	226.94	6.13
CAP CONC PIER-5	226.94	228.38	1.45
SHAFT FORM PIER-4	222.33	233.15	10.82
SHAFT REBAR PIER-4	233.15	239.79	6.64
SHAFT CONC PIER-4	239.79	240.94	1.15
SHAFT FORM PIER-42	241.94	247.01	5.06
SHAFT REBAR PIER-42	247.01	255.88	8.87
SHAFT CONC PIER-42	255.88	257.01	1.14
CAP FORM PIER-4	257.01	262.55	5.54
CAP REBAR PIER-4	262.55	267.83	5.28
CAP CONC PIER-4	267.83	271.60	3.77

SLAMSYSTEM Input file for the basecase scenario

```
1 GEN,ANIL SAWHNEY,PEACE RIVER BRIDGE,4/22/1994,1,Y,Y,Y/Y,Y,Y/1,72;
2 LIMITS,123,10,2000;
3 INITIALIZE,,,Y;
4 NETWORK;
5 ;FILE BERMWST.NET, NODE LABEL SEED LAAA
6 ;FILE BLINDP1.NET, NODE LABEL SEED WAAA
7 ;FILE BLINDP2.NET, NODE LABEL SEED JAAA
8 ;FILE CAPCONP1.NET, NODE LABEL SEED MAAA
9 ;FILE CAPFORP1.NET, NODE LABEL SEED OAAA
10 ;FILE CAPREBP1.NET, NODE LABEL SEED NAAA
11 ;FILE COFERP3.NET, NODE LABEL SEED KAAA
12 ;FILE EXCP1.NET, NODE LABEL SEED YAAA
13 ;FILE FTCONP1.NET, NODE LABEL SEED SAAA
14 ;FILE FTFP1.NET, NODE LABEL SEED VAAA
15 ;FILE P1FOTF.NET, NODE LABEL SEED IAAA
16 ;FILE P2CAPC.NET, NODE LABEL SEED ADAA
17 ;FILE P2CAPF.NET, NODE LABEL SEED ABAA
18 ;FILE P2CAPR.NET, NODE LABEL SEED ACAA
19 ;FILE P2COFD.NET, NODE LABEL SEED TTAA
20 ;FILE P2FOTC.NET, NODE LABEL SEED ZAAA
21 ;FILE P2FOTR.NET, NODE LABEL SEED HAAA
22 ;FILE P2PEDC.NET, NODE LABEL SEED DAAA
23 ;FILE P2PEDF.NET, NODE LABEL SEED FAAA
24 ;FILE P2PEDR.NET, NODE LABEL SEED EAAA
25 ;FILE P2SHC.NET, NODE LABEL SEED AAAA
26 ;FILE P2SHF.NET, NODE LABEL SEED CAAA
27 ;FILE P2SHR.NET, NODE LABEL SEED BAAA
28 ;FILE P3BLIND.NET, NODE LABEL SEED ADAA
29 ;FILE P3CAPC.NET, NODE LABEL SEED AOAA
30 ;FILE P3CAPF.NET, NODE LABEL SEED AMAA
31 ;FILE P3CAPR.NET, NODE LABEL SEED ANAA
32 ;FILE P3FOTC.NET, NODE LABEL SEED AGAA
33 ;FILE P3FOTF.NET, NODE LABEL SEED AEAA
34 ;FILE P3FOTR.NET, NODE LABEL SEED AFAA
35 ;FILE P3PEDC.NET, NODE LABEL SEED AIAA
36 ;FILE P3PEDF.NET, NODE LABEL SEED AHAA
37 ;FILE P3PEDR.NET, NODE LABEL SEED AIAA
38 ;FILE P3SHC.NET, NODE LABEL SEED ALAA
39 ;FILE P3SHF.NET, NODE LABEL SEED AJAA
40 ;FILE P3SHR.NET, NODE LABEL SEED AKAA
41 ;FILE P4BERM.NET, NODE LABEL SEED LPAA
42 ;FILE P4BLIND.NET, NODE LABEL SEED TLAA
43 ;FILE P4CAPC.NET, NODE LABEL SEED TRUA
44 ;FILE P4CAPF.NET, NODE LABEL SEED XUAA
45 ;FILE P4CAPR.NET, NODE LABEL SEED LMAA
46 ;FILE P4COFD.NET, NODE LABEL SEED MKAA
47 ;FILE P4FOTC.NET, NODE LABEL SEED NNNA
48 ;FILE P4FOTF.NET, NODE LABEL SEED UUAA
49 ;FILE P4FOTR.NET, NODE LABEL SEED MMMA
50 ;FILE P4PEDC.NET, NODE LABEL SEED TPAA
51 ;FILE P4PEDF.NET, NODE LABEL SEED TUAA
52 ;FILE P4PEDR.NET, NODE LABEL SEED TYAA
53 ;FILE P4SHC.NET, NODE LABEL SEED BXAA
```

```
54  ;FILE P4SHF.NET, NODE LABEL SEED LBAA
55  ;FILE P4SHR.NET, NODE LABEL SEED LUAA
56  ;FILE P5BERM.NET, NODE LABEL SEED LLAA
57  ;FILE P5BLIND.NET, NODE LABEL SEED ZZZA
58  ;FILE P5CAPC.NET, NODE LABEL SEED NYAA
59  ;FILE P5CAPF.NET, NODE LABEL SEED MOAA
60  ;FILE P5CAPR.NET, NODE LABEL SEED OMAA
61  ;FILE P5COFD.NET, NODE LABEL SEED KXYA
62  ;FILE P5FOTC.NET, NODE LABEL SEED MMMA
63  ;FILE P5FOTF.NET, NODE LABEL SEED KKKA
64  ;FILE P5FOTR.NET, NODE LABEL SEED LLLA
65  ;FILE P5PEDC.NET, NODE LABEL SEED NTAA
66  ;FILE P5PEDF.NET, NODE LABEL SEED IIAA
67  ;FILE P5PEDR.NET, NODE LABEL SEED LTAA
68  ;FILE P5SHC.NET, NODE LABEL SEED ITAA
69  ;FILE P5SHF.NET, NODE LABEL SEED TIAA
70  ;FILE P5SHR.NET, NODE LABEL SEED MEAA
71  ;FILE P6BLIND.NET, NODE LABEL SEED ZZZZ
72  ;FILE P6CAPC.NET, NODE LABEL SEED KNAA
73  ;FILE P6CAPF.NET, NODE LABEL SEED KMAA
74  ;FILE P6CAPR.NET, NODE LABEL SEED LKAA
75  ;FILE P6FOTC.NET, NODE LABEL SEED ZXAA
76  ;FILE P6FOTF.NET, NODE LABEL SEED ZZZB
77  ;FILE P6FOTR.NET, NODE LABEL SEED ZZAA
78  ;FILE P6PEDC.NET, NODE LABEL SEED XXAA
79  ;FILE P6PEDF.NET, NODE LABEL SEED ZYAA
80  ;FILE P6PEDR.NET, NODE LABEL SEED ZTAA
81  ;FILE P6SHC.NET, NODE LABEL SEED KKAA
82  ;FILE P6SHF.NET, NODE LABEL SEED XYAA
83  ;FILE P6SHR.NET, NODE LABEL SEED XZAA
84  ;FILE PILAB1.NET, NODE LABEL SEED ZAAA
85  ;FILE PILPR1.NET, NODE LABEL SEED ZAAA
86  ;FILE REBFTP1.NET, NODE LABEL SEED TAAA
87  ;FILE REBSHP1.NET, NODE LABEL SEED QAAA
88  ;FILE SHCONP1.NET, NODE LABEL SEED PAAA
89  ;FILE SHFORP1.NET, NODE LABEL SEED RAAA
90  ;FILE START.NET, NODE LABEL SEED ACAA
91      RESOURCE/1,CRANE40T,1,4;
92      RESOURCE/2,DISELHAM,1,4,41,83,85,87;
93      RESOURCE/3,PILCREW,1,4,41,83,85,87;
94      RESOURCE/4,EXCVATOR,3,2;
95      RESOURCE/5,TRUCKWST(5),3,19;
96      RESOURCE/6,CONPMPWT,6,10,14,18,23,27,31,35,39,43,47,51,55,59;
97      RESOURCE/7,TRMIXRWT(2),5,9,13,17,22,26,30,34,38,42,46,50,54,58;
98      RESOURCE/8,CONCRWST,6,10,14,18,23,27,31,35,39,43,47,51,55,59;
99      RESOURCE/9,FOTFORWT,7,24,44;
100     RESOURCE/10,CRANEFWT,7,11,15,24,28,32,36,44,48,52,56;
101     RESOURCE/11,FORCRWST,7,11,15,24,28,32,36,44,48,52,56;
102     RESOURCE/12,CRANERB,8,12,16,25,29,33,37,40,45,49,53,57,82;
103     RESOURCE/13,RBCRWST,8,12,16,25,29,33,37,45,49,53,57;
104     RESOURCE/14,SHFTFRWT,11,32,52;
105     RESOURCE/15,CAPFRMWT,15,36,56;
106     RESOURCE/16,DOZERWST,20;
107     RESOURCE/17,COFSTC(4),40,82,84,86;
```

```
108      RESOURCE/18,PEDFRMWT,28,48;
109      RESOURCE/19,TRUCKEST(5),78,80;
110      RESOURCE/20,CONPMPET,61,65,69,73,77,89,93,97,101,105,107,111,115,119,123;
111      RESOURCE/21,TRMIXRET(2),60,64,68,72,76,88,92,96,100,104,106,110,114,118,112,122;
113      RESOURCE/22,CONCREST,61,65,69,73,77,89,93,97,101,105,107,111,115,119,123;
114      RESOURCE/23,FOTFORET,62,90,108;
115      RESOURCE/24,CRANEFET,62,66,70,74,90,94,98,102,108,112,116,120;
116      RESOURCE/25,FORCREST,62,66,70,74,90,94,98,102,108,112,116,120;
117      RESOURCE/26,CRANERBE,63,67,71,75,84,86,91,99,103,109,113,117,121;
118      RESOURCE/27,RBCREST,63,67,71,75,91,95,99,103,109,113,117,121;
119      RESOURCE/28,SHFTFRET,70,98,116;
120      RESOURCE/29,CAPFRMET,74,102,120;
121      RESOURCE/30,DOZEREST,79,81;
122      RESOURCE/31,PEDFRMET,66,94,112;
123  ;FILE BERMWST.NET, NODE LABEL SEED LAAA
124  ;
125  BRWT  EVENT,6,1;
126      ACTIVITY;
127  BWA   AWAIT(19),TRUCKWST,,1;
128      ACTIVITY/29,30,,;DELIVDIRTW;
129      FREE,TRUCKWST,1;
130      ACTIVITY;
131  BWB   AWAIT(20),DOZERWST,,1;
132      ACTIVITY/30,20,,;DOZBRWT;
133      FREE,DOZERWST;
134      ACTIVITY;
135      ACCUMULATE,800,800,,2;
136      ACTIVITY;
137      ACTIVITY,,,COP3;
138      TERMINATE;
139  ;FILE BLINDP1.NET, NODE LABEL SEED WAAA
140  ;
141  BLP1  EVENT,2,1;
142      ACTIVITY;
143  ACON  AWAIT(5),TRMIXRWT,,1;
144      ACTIVITY/12,15,,;DELIVER CONC;
145      FREE,TRMIXRWT,1;
146      ACTIVITY;
147  CRED  AWAIT(6),ALLOC(4),,1;
148      ACTIVITY/13,10,,;PLACE CONC;
149      FREE,CONPMPWT,1;
150      ACTIVITY;
151      FREE,CONCRWST,1;
152      ACTIVITY;
153      ACCUMULATE,5,5,,2;
154      ACTIVITY,,,FFP1;
155  ;FILE BLINDP2.NET, NODE LABEL SEED JAAA
156  ;
157  BLP2  EVENT,7,1;
158      ACTIVITY;
159  P2B1  AWAIT(22),TRMIXRWT,,1;
160      ACTIVITY/35,15,,;DELV CONC BP
161      FREE,TRMIXRWT,1;
162      ACTIVITY;
```

```
163      AWAIT(23),ALLOC(16),,1;
164      ACTIVITY/36,10,,;PLACE CON BP
165      FREE,CONPMPWT,1;
166      ACTIVITY;
167      FREE,CONCRWST,1;
168      ACTIVITY;
169      ACCUMULATE,5,5,,1;
170      ACTIVITY,,,FFP2;
171  ;FILE CAPCONP1.NET, NODE LABEL SEED MAAA
172  ;
173  CAPC  EVENT,5,1;
174      ACTIVITY;
175  CAPA  AWAIT(17),TRMIXRWT,,1;
176      ACTIVITY/26,15,,;DELCON CAP P1;
177      FREE,TRMIXRWT,1;
178      ACTIVITY;
179  CAPB  AWAIT(18),ALLOC(13),,1;
180      ACTIVITY/27,10,,;PLACECONCAPP1;
181      FREE,CONPMPWT,1;
182      ACTIVITY;
183      FREE,CONCRWST,1;
184      ACTIVITY;
185      ACCUMULATE,40,40,,2;
186      ACTIVITY/28,480,,;REL CAPFORM;
187      FREE,CAPFRMWT;
188      ACTIVITY,,,CDP2;
189  ;FILE CAPFORP1.NET, NODE LABEL SEED OAAA
190  ;
191  CAPF  AWAIT(15),ALLOC(11),,1;
192      ACTIVITY/24,1440,,;FORMCAPP1;
193      GOON,3;
194      ACTIVITY,,,CAPR;
195      ACTIVITY;
196      ACTIVITY,,,OAAB;
197      FREE,FORCRWST,1;
198      TERMINATE;
199  OAAB  FREE,CRANEFWT,1;
200      TERMINATE;
201  ;FILE CAPREBP1.NET, NODE LABEL SEED NAAA
202  ;
203  CAPR  AWAIT(16),ALLOC(12),,1;
204      ACTIVITY/25,2880,,;REBCAPP1;
205      GOON,3;
206      ACTIVITY,,,CAPC;
207      ACTIVITY;
208      ACTIVITY,,,NAAB;
209      FREE,CRANERB,1;
210      TERMINATE;
211  NAAB  FREE,RBCRWST,1;
212      TERMINATE;
213  ;FILE COFERP3.NET, NODE LABEL SEED KAAA
214  ;
215  COP3  AWAIT(40),ALLOC(14),,1;
216      ACTIVITY/56,240,,;FLOATCAGEP3;
```

```
217      GOON,1;
218      ACTIVITY/57,240,,;POS&ANCHP3;
219 COFB  AWAIT(41),ALLOC(15),,1;
220      ACTIVITY/58,480,,;DRIVE SPUDPILES;
221 KAAB  GOON,1;
222      ACTIVITY/59,360,,;DRIVE SHEET COF;
223      ASSIGN,ATRIB(3)=ATRIB(3) + 1,1;
224      ACTIVITY,,ATRIB(3).GE.100;
225      ACTIVITY,,ATRIB(3).LT.100,KAAB;
226      FREE,CRANERB,1;
227      ACTIVITY;
228      FREE,DISELHAM,1;
229      ACTIVITY;
230      FREE,PILCREW;
231      ACTIVITY,,,BLP3;
232 ;FILE EXCP1.NET, NODE LABEL SEED YAAA
233 ;
234 CEXC  CREATE,0,,,464,1;
235      ACTIVITY;
236 AWEX  AWAIT(2),EXCVATOR,,1;
237      ACTIVITY/5,3,,;EXCAVATE;
238      FREE,EXCVATOR,1;
239      ACTIVITY;
240 TRKA  ACCUMULATE,8,8,,1;
241      ACTIVITY;
242 LOAD  AWAIT(3),ALLOC(2),,1;
243      ACTIVITY/6,3,,;LOAD TRUCK;
244 FEXC  FREE,EXCVATOR,1;
245      ACTIVITY/7,60,,;TRAVEL&DUMP;
246 FTRK  FREE,TRUCKWST,1;
247      ACTIVITY;
248 TOT   ACCUMULATE,58,58,,2;
249      ACTIVITY,,,PLP1;
250      ACTIVITY,,,BRWT;
251 ;FILE FTCONP1.NET, NODE LABEL SEED SAAA
252 ;
253 CFP1  EVENT,3,1;
254      ACTIVITY;
255 ACFP  AWAIT(9),TRMIXRWT,,1;
256      ACTIVITY/16,15,,;DEL CON FT P1;
257      FREE,TRMIXRWT,1;
258      ACTIVITY;
259 CFAP  AWAIT(10),ALLOC(7),,1;
260      ACTIVITY/17,10,,;PLACECONCFTP1;
261      FREE,CONPMPWT,1;
262      ACTIVITY;
263      FREE,CONCRWST,1;
264      ACTIVITY;
265      ACCUMULATE,80,80,,2;
266      ACTIVITY/18,480,,;REL FOTFORM;
267      ACTIVITY,,,SHP1;
268      FREE,FOTFORWT;
269      TERMINATE;
270 ;FILE FTFP1.NET, NODE LABEL SEED VAAA
```

```
271 ;
272 FFP1  AWAIT(7),ALLOC(5),,1;
273       ACTIVITY/14,1440,,;FORMFTP1;
274       GOON,3;
275       ACTIVITY;
276       ACTIVITY,,,VAAB;
277       ACTIVITY,,,RFP1;
278       FREE,FORCRWST,1;
279       TERMINATE;
280 VAAB  FREE,CRANEFWT,1;
281       TERMINATE;
282 ;FILE P1FOTF.NET, NODE LABEL SEED IAAA
283 ;
284 FFP2  AWAIT(24),ALLOC(17),,1;
285       ACTIVITY/37,1440,,;FORMFTP2;
286       GOON,3;
287       ACTIVITY,,,RFP2;
288       ACTIVITY;
289       ACTIVITY,,,IAAB;
290       FREE,FORCRWST,1;
291       ACTIVITY;
292       TERMINATE;
293 IAAB  FREE,CRANEFWT,1;
294       ACTIVITY;
295       TERMINATE;
296 ;FILE P2CAPC.NET, NODE LABEL SEED ADAA
297 ;
298 CCP2  EVENT,11,1;
299       ACTIVITY;
300 CCPA  AWAIT(38),TRMIXRWT,,1;
301       ACTIVITY/53,15,,;DELCON CAP P2;
302       FREE,TRMIXRWT,1;
303       ACTIVITY;
304 CP2B  AWAIT(39),ALLOC(28),,1;
305       ACTIVITY/54,10,,;PLACECONCAPP2;
306       FREE,CONPMPWT,1;
307       ACTIVITY;
308       FREE,CONCRWST,1;
309       ACTIVITY;
310       ACCUMULATE,40,40,,2;
311       ACTIVITY/55,480,,;REL CAPFORM P2;
312       FREE,CAPFRMWT;
313       TERMINATE;
314 ;FILE P2CAPF.NET, NODE LABEL SEED ABAA
315 ;
316 CAP2  AWAIT(36),ALLOC(26),,1;
317       ACTIVITY/51,1440,,;FORMCAPP2;
318       GOON,3;
319       ACTIVITY;
320       ACTIVITY,,,CRP2;
321       ACTIVITY,,,ABAB;
322       FREE,FORCRWST,1;
323       ACTIVITY;
324       TERMINATE;
```

```
ABAB  FREE,CRANEFWT,1;
      ACTIVITY;
      TERMINATE;
;FILE P2CAPR.NET, NODE LABEL SEED ACAA
;
CRP2  AWAIT(37),ALLOC(27),,1;
      ACTIVITY/52,2880,,;REBCAPP2;
      GOON,3;
      ACTIVITY,,,CCP2;
      ACTIVITY;
      ACTIVITY,,,ACAB;
      FREE,CRANERB,1;
      TERMINATE;
ACAB  FREE,RBCRWST,1;
      TERMINATE;
;FILE P2COFD.NET, NODE LABEL SEED TTAA
;
CDP2  AWAIT(82),ALLOC(56),,1;
      ACTIVITY/108,240,,;FLOATCAGEP2;
      GOON,1;
      ACTIVITY/109,240,,;POS&ANCHP2;
CP21  AWAIT(83),ALLOC(57),,1;
      ACTIVITY,480,,;DRVSPDPILP2;
TTAD  GOON,1;
      ACTIVITY/110,360,,;DRIVSHTCOFP2;
      ASSIGN,ATRIB(6)=ATRIB(6) + 1,1;
      ACTIVITY,,ATRIB(6).GE.100;
      ACTIVITY,,ATRIB(6).LT.100,TTAD;
      FREE,CRANERB,1;
      ACTIVITY;
      FREE,DISELHAM,1;
      ACTIVITY;
      FREE,PILCREW;
      ACTIVITY,,,BLP2;
;FILE P2FOTC.NET, NODE LABEL SEED ZAAA
;
CFP2  EVENT,8,1;
      ACTIVITY;
P2A   AWAIT(26),TRMIXRWT,,1;
      ACTIVITY/39,15,,;DELCON FTP2;
      FREE,TRMIXRWT,1;
      ACTIVITY;
P2B   AWAIT(27),ALLOC(19),,1;
      ACTIVITY,10;
      FREE,CONPMPWT,1;
      ACTIVITY;
      FREE,CONCRWST,1;
      ACTIVITY;
      ACCUMULATE,80,80,,2;
      ACTIVITY/40,480,,;REL FOTFORM;
      ACTIVITY,,,PDP2;
      FREE,FOTFORWT;
      ACTIVITY;
      TERMINATE;
```

```
;FILE P2FOTR.NET, NODE LABEL SEED HAAA
;
RFP2  AWAIT(25),ALLOC(18),,1;
      ACTIVITY/38,2880,,;REBFTP2;
      GOON,3;
      ACTIVITY;
      ACTIVITY,,,CFP2;
      ACTIVITY,,,HAAB;
      FREE,CRANERB,1;
      ACTIVITY;
      TERMINATE;
HAAB  FREE,RBCRWST,1;
      ACTIVITY;
      TERMINATE;
;FILE P2PEDC.NET, NODE LABEL SEED DAAA
;
P2CP  EVENT,9,1;
      ACTIVITY;
APP2  AWAIT(30),TRMIXRWT,,1;
      ACTIVITY/43,15,,;DELCON PED P
      FREE,TRMIXRWT,1;
      ACTIVITY;
CPP2  AWAIT(31),ALLOC(22),,1;
      ACTIVITY/44,10,,;PLACCONPEDP2
      FREE,CONPMPWT,1;
      ACTIVITY;
      FREE,CONCRWST,1;
      ACTIVITY;
      ACCUMULATE,80,80,,2;
      ACTIVITY/45,480,,;REL PEDFORM;
      ACTIVITY,,,P2SH;
      FREE,PEDFRMWT;
      ACTIVITY;
      TERMINATE;
;FILE P2PEDF.NET, NODE LABEL SEED FAAA
;
PDP2  AWAIT(28),ALLOC(20),,1;
      ACTIVITY/41,1440,,;FRMPEDP2;
      GOON,3;
      ACTIVITY,,,RPED;
      ACTIVITY;
      ACTIVITY,,,FAAB;
      FREE,FORCRWST,1;
      ACTIVITY;
      TERMINATE;
FAAB  FREE,CRANEFWT,1;
      ACTIVITY;
      TERMINATE;
;FILE P2PEDR.NET, NODE LABEL SEED EAAA
;
RPED  AWAIT(29),ALLOC(21),,1;
      ACTIVITY/42,2880,,;REBPEDP2;
      GOON,3;
      ACTIVITY;
```

```
433      ACTIVITY,,,EAAB;
434      ACTIVITY,,,P2CP;
435      FREE,CRANERB,1;
436      ACTIVITY;
437      TERMINATE;
438 EAAB  FREE,RBCRWST,1;
439      ACTIVITY;
440      TERMINATE;
441 ;FILE P2SHC.NET, NODE LABEL SEED AAAA
442 ;
443 CSP2  EVENT,10,1;
444      ACTIVITY;
445 ACSH  AWAIT(34),TRMIXRWT,,1;
446      ACTIVITY/48,15,,;DEL CON SH P
447      FREE,TRMIXRWT,1;
448      ACTIVITY;
449 CSHP  AWAIT(35),ALLOC(25),,1;
450      ACTIVITY/49,10,,;PLACECONCSHP
451      FREE,CONPMPWT,1;
452      ACTIVITY;
453      FREE,CONCRWST,1;
454      ACTIVITY;
455      ACCUMULATE,76,76,,2;
456      ACTIVITY/50,480,,;RELSHTFORM P
457      ACTIVITY,,,AAAB;
458      FREE,SHFTFRWT;
459      ACTIVITY;
460      TERMINATE;
461 AAAB  ASSIGN,ATRIB(4)=ATRIB(4) + 1,1;
462      ACTIVITY,,ATRIB(4).EQ.1,P2SH;
463      ACTIVITY,,ATRIB(4).EQ.2,CAP2;
464 ;FILE P2SHF.NET, NODE LABEL SEED CAAA
465 ;
466 P2SH  AWAIT(32),ALLOC(23),,1;
467      ACTIVITY/46,1440,,;FRMSHFTP2;
468      GOON,3;
469      ACTIVITY;
470      ACTIVITY,,,CAAB;
471      ACTIVITY,,,RSP2;
472      FREE,FORCRWST,1;
473      ACTIVITY;
474      TERMINATE;
475 CAAB  FREE,CRANEFWT,1;
476      ACTIVITY;
477      TERMINATE;
478 ;FILE P2SHR.NET, NODE LABEL SEED BAAA
479 ;
480 RSP2  AWAIT(33),ALLOC(24),,1;
481      ACTIVITY/47,2880,,;REBSHP2;
482      GOON,3;
483      ACTIVITY;
484      ACTIVITY,,,BAAB;
485      ACTIVITY,,,CSP2;
486      FREE,CRANERB,1;
```

```
487      ACTIVITY;
488      TERMINATE;
489 BAAB  FREE,RBCRWST,1;
490      ACTIVITY;
491      TERMINATE;
492 ;FILE P3BLIND.NET, NODE LABEL SEED ADAA
493 ;
494 BLP3  EVENT,12,1;
495      ACTIVITY;
496 P3B1  AWAIT(42),TRMIXRWT,,1;
497      ACTIVITY/60,15,,;DELCONCBP3
498      FREE,TRMIXRWT,1;
499      ACTIVITY;
500      AWAIT(43),ALLOC(29),,1;
501      ACTIVITY/61,10,,;PLACCONBP3
502      FREE,CONPMPWT,1;
503      ACTIVITY;
504      FREE,CONCRWST,1;
505      ACTIVITY;
506      ACCUMULATE,5,5,,1;
507      ACTIVITY,,,FFP3;
508 ;FILE P3CAPC.NET, NODE LABEL SEED AOAA
509 ;
510 CCP3  EVENT,16,1;
511      ACTIVITY;
512 P3CA  AWAIT(58),TRMIXRWT,,1;
513      ACTIVITY/79,15,,;DELCONCAPP3;
514      FREE,TRMIXRWT,1;
515      ACTIVITY;
516 P3B   AWAIT(59),ALLOC(41),,1;
517      ACTIVITY/80,10,,;PLCONCAPP3;
518      FREE,CONPMPWT,1;
519      ACTIVITY;
520      FREE,CONCRWST,1;
521      ACTIVITY;
522      ACCUMULATE,40,40,,2;
523      ACTIVITY/81,480,,;RELCAPFORMP3;
524      FREE,CAPFRMWT;
525      TERMINATE;
526 ;FILE P3CAPF.NET, NODE LABEL SEED AMAA
527 ;
528 C1P3  AWAIT(56),ALLOC(39),,1;
529      ACTIVITY/77,1440,,;FORMCAPP3;
530      GOON,3;
531      ACTIVITY;
532      ACTIVITY,,,CRP3;
533      ACTIVITY,,,AMAB;
534      FREE,FORCRWST,1;
535      ACTIVITY;
536      TERMINATE;
537 AMAB  FREE,CRANEFWT,1;
538      ACTIVITY;
539      TERMINATE;
540 ;FILE P3CAPR.NET, NODE LABEL SEED ANAA
```

```
541 ;
542 CRP3  AWAIT(57),ALLOC(40),,1;
543       ACTIVITY/78,2880,,;REBCAPP3;
544       GOON,3;
545       ACTIVITY,,,CCP3;
546       ACTIVITY;
547       ACTIVITY,,,ANAD;
548       FREE,CRANERB,1;
549       TERMINATE;
550 ANAD  FREE,RBCRWST,1;
551       TERMINATE;
552 ;FILE P3FOTC.NET, NODE LABEL SEED AGAA
553 ;
554 CFP3  EVENT,13,1;
555       ACTIVITY;
556 P31   AWAIT(46),TRMIXRWT,,1;
557       ACTIVITY/64,15,,;DELCONFTP3;
558       FREE,TRMIXRWT,1;
559       ACTIVITY;
560 P32   AWAIT(47),ALLOC(32),,1;
561       ACTIVITY,10;
562       FREE,CONPMPWT,1;
563       ACTIVITY;
564       FREE,CONCRWST,1;
565       ACTIVITY;
566       ACCUMULATE,80,80,,2;
567       ACTIVITY/66,480,,;RELFOTFRMP3;
568       ACTIVITY,,,PFP3;
569       FREE,FOTFORWT;
570       ACTIVITY;
571       TERMINATE;
572 ;FILE P3FOTF.NET, NODE LABEL SEED AEAA
573 ;
574 FFP3  AWAIT(44),ALLOC(30),,1;
575       ACTIVITY/62,1440,,;FORMFTP3;
576       GOON,3;
577       ACTIVITY,,,RFP3;
578       ACTIVITY;
579       ACTIVITY,,,AEAB;
580       FREE,FORCRWST,1;
581       ACTIVITY;
582       TERMINATE;
583 AEAB  FREE,CRANEFWT,1;
584       ACTIVITY;
585       TERMINATE;
586 ;FILE P3FOTR.NET, NODE LABEL SEED AFAA
587 ;
588 RFP3  AWAIT(45),ALLOC(31),,1;
589       ACTIVITY/63,2880,,;REBFTP3;
590       GOON,3;
591       ACTIVITY;
592       ACTIVITY,,,CFP3;
593       ACTIVITY,,,AFAB;
594       FREE,CRANERB,1;
```

```
595      ACTIVITY;
596      TERMINATE;
597  AFAB  FREE,RBCRWST,1;
598      ACTIVITY;
599      TERMINATE;
600  ;FILE P3PEDC.NET, NODE LABEL SEED AIAA
601  ;
602  PCP3  EVENT,14,1;
603      ACTIVITY;
604  APP3  AWAIT(50),TRMIXRWT,,1;
605      ACTIVITY/69,15,,;DELCONPEDP3;
606      FREE,TRMIXRWT,1;
607      ACTIVITY;
608  CPP3  AWAIT(51),ALLOC(35),,1;
609      ACTIVITY/70,10,,;PLCONPEDP3;
610      FREE,CONPMPWT,1;
611      ACTIVITY;
612      FREE,CONCRWST,1;
613      ACTIVITY;
614      ACCUMULATE,80,80,,2;
615      ACTIVITY/71,480,,;RLPEDFRMP3;
616      ACTIVITY,,,SFP3;
617      FREE,PEDFRMWT;
618      ACTIVITY;
619      TERMINATE;
620  ;FILE P3PEDF.NET, NODE LABEL SEED AHAA
621  ;
622  PFP3  AWAIT(48),ALLOC(33),,1;
623      ACTIVITY/67,1440,,;FRMPEDP3;
624      GOON,3;
625      ACTIVITY,,,PRP3;
626      ACTIVITY;
627      ACTIVITY,,,AHAB;
628      FREE,FORCRWST,1;
629      ACTIVITY;
630      TERMINATE;
631  AHAB  FREE,CRANEFWT,1;
632      ACTIVITY;
633      TERMINATE;
634  ;FILE P3PEDR.NET, NODE LABEL SEED AIAA
635  ;
636  PRP3  AWAIT(49),ALLOC(34),,1;
637      ACTIVITY/42,2880,,;REBPEDP2;
638      GOON,3;
639      ACTIVITY;
640      ACTIVITY,,,AIAB;
641      ACTIVITY,,,PCP3;
642      FREE,CRANERB,1;
643      ACTIVITY;
644      TERMINATE;
645  AIAB  FREE,RBCRWST,1;
646      ACTIVITY;
647      TERMINATE;
648  ;FILE P3SHC.NET, NODE LABEL SEED ALAA
```

```
649 ;
650 SCP3  EVENT,15,1;
651       ACTIVITY;
652 SCPA  AWAIT(54),TRMIXRWT,,1;
653       ACTIVITY/74,15,,;DELCONSHP3;
654       FREE,TRMIXRWT,1;
655       ACTIVITY;
656 SCPB  AWAIT(55),ALLOC(38),,1;
657       ACTIVITY/75,10,,;PLCCONSHP3;
658       FREE,CONPMPWT,1;
659       ACTIVITY;
660       FREE,CONCRWST,1;
661       ACTIVITY;
662       ACCUMULATE,76,76,,2;
663       ACTIVITY/76,480,,;RELSHFRMP3;
664       ACTIVITY,,,ALAB;
665       FREE,SHFTFRWT;
666       ACTIVITY;
667       TERMINATE;
668 ALAB  ASSIGN,ATRIB(5)=ATRIB(5) + 1,1;
669       ACTIVITY,,ATRIB(5).EQ.1,SFP3;
670       ACTIVITY,,ATRIB(5).EQ.2,C1P3;
671 ;FILE P3SHF.NET, NODE LABEL SEED AJAA
672 ;
673 SFP3  AWAIT(52),ALLOC(36),,1;
674       ACTIVITY/72,1440,,;FRMSHFTP3;
675       GOON,3;
676       ACTIVITY;
677       ACTIVITY,,,AJAC;
678       ACTIVITY,,,SRP3;
679       FREE,FORCRWST,1;
680       ACTIVITY;
681       TERMINATE;
682 AJAC  FREE,CRANEFWT,1;
683       ACTIVITY;
684       TERMINATE;
685 ;FILE P3SHR.NET, NODE LABEL SEED AKAA
686 ;
687 SRP3  AWAIT(53),ALLOC(37),,1;
688       ACTIVITY/73,2880,,;REBSHP3;
689       GOON,3;
690       ACTIVITY;
691       ACTIVITY,,,AKAB;
692       ACTIVITY,,,SCP3;
693       FREE,CRANERB,1;
694       ACTIVITY;
695       TERMINATE;
696 AKAB  FREE,RBCRWST,1;
697       ACTIVITY;
698       TERMINATE;
699 ;FILE P4BERM.NET, NODE LABEL SEED LPAA
700 ;
701 BRP4  EVENT,23,1;
702       ACTIVITY;
```

```
703 BP41  AWAIT(80),TRUCKEST,,1;
704       ACTIVITY/106,30,,;DELVDRTP4;
705       FREE,TRUCKEST,1;
706       ACTIVITY;
707 BP42  AWAIT(81),DOZEREST,,1;
708       ACTIVITY/107,20,,;DOZBP4;
709       FREE,DOZEREST;
710       ACTIVITY;
711       ACCUMULATE,800,800,,2;
712       ACTIVITY;
713       ACTIVITY,,,CDP4;
714       TERMINATE;
715 ;FILE P4BLIND.NET, NODE LABEL SEED TLAA
716 ;
717 BLP4  EVENT,29,1;
718       ACTIVITY;
719 P4B1  AWAIT(106),TRMIXRET,,1;
720       ACTIVITY/138,15,,;DELCONCBP4;
721       FREE,TRMIXRET,1;
722       ACTIVITY;
723       AWAIT(107),ALLOC(75),,1;
724       ACTIVITY/139,10,,;PLACCONBP4;
725       FREE,CONPMPET,1;
726       ACTIVITY;
727       FREE,CONCREST,1;
728       ACTIVITY;
729       ACCUMULATE,5,5,,1;
730       ACTIVITY,,,FFP4;
731 ;FILE P4CAPC.NET, NODE LABEL SEED TRUA
732 ;
733 C1P4  EVENT,33,1;
734       ACTIVITY;
735 P4CA  AWAIT(122),TRMIXRET,,1;
736       ACTIVITY/157,15,,;DELCONCAPP4;
737       FREE,TRMIXRET,1;
738       ACTIVITY;
739 P44   AWAIT(123),ALLOC(87),,1;
740       ACTIVITY/158,10,,;PLCONCAPP4;
741       FREE,CONPMPET,1;
742       ACTIVITY;
743       FREE,CONCREST,1;
744       ACTIVITY;
745       ACCUMULATE,40,40,,2;
746       ACTIVITY/159,480,,;RELCAPFORMP4;
747       FREE,CAPFRMET;
748       TERMINATE;
749 ;FILE P4CAPF.NET, NODE LABEL SEED XUAA
750 ;
751 FCP4  AWAIT(120),ALLOC(85),,1;
752       ACTIVITY/155,1440,,;FORMCAPP4;
753       GOON,3;
754       ACTIVITY;
755       ACTIVITY,,,RCP4;
756       ACTIVITY,,,XUAD;
```

```
757      FREE,FORCREST,1;
758      ACTIVITY;
759      TERMINATE;
760 XUAD  FREE,CRANEFET,1;
761      ACTIVITY;
762      TERMINATE;
763 ;FILE P4CAPR.NET, NODE LABEL SEED LMAA
764 ;
765 RCP4  AWAIT(121),ALLOC(86),,1;
766      ACTIVITY/156,2880,,;REBCAPP4;
767      GOON,3;
768      ACTIVITY,,,C1P4;
769      ACTIVITY;
770      ACTIVITY,,,LMAC;
771      FREE,CRANERBE,1;
772      TERMINATE;
773 LMAC  FREE,RBCREST,1;
774      TERMINATE;
775 ;FILE P4COFD.NET, NODE LABEL SEED MKAA
776 ;
777 CDP4  AWAIT(86),ALLOC(60),,1;
778      ACTIVITY/114,240,,;FLOATCAGEP4;
779      GOON,1;
780      ACTIVITY/115,240,,;POS&ANCHP4;
781 C4PA  AWAIT(87),ALLOC(61),,1;
782      ACTIVITY,480,,;DRIVSPUDPILP4;
783 MKAB  GOON,1;
784      ACTIVITY/116,360,,;DRIV SHETP4;
785      ASSIGN,ATRIB(8)=ATRIB(8) + 1,1;
786      ACTIVITY,,ATRIB(8).GE.100;
787      ACTIVITY,,ATRIB(8).LT.100,MKAB;
788      FREE,CRANERBE,1;
789      ACTIVITY;
790      FREE,DISELHAM,1;
791      ACTIVITY;
792      FREE,PILCREW;
793      ACTIVITY,,,BLP4;
794 ;FILE P4FOTC.NET, NODE LABEL SEED NNNA
795 ;
796 CFP4  EVENT,30,1;
797      ACTIVITY;
798 P41   AWAIT(110),TRMIXRET,,1;
799      ACTIVITY/142,15,,;DELCONFTP4;
800      FREE,TRMIXRET,1;
801      ACTIVITY;
802 P42   AWAIT(111),ALLOC(78),,1;
803      ACTIVITY/143,10,,;PLCCONP4;
804      FREE,CONPMPET,1;
805      ACTIVITY;
806      FREE,CONCREST,1;
807      ACTIVITY;
808      ACCUMULATE,80,80,,2;
809      ACTIVITY/144,480,,;RELFOTFRMP4;
810      ACTIVITY,,,PFP4;
```

```
811      FREE,FOTFORET;
812      ACTIVITY;
813      TERMINATE;
814  ;FILE P4FOTF.NET, NODE LABEL SEED UUAA
815  ;
816  FFP4  AWAIT(108),ALLOC(76),,1;
817      ACTIVITY/140,1440,,;FORMFTP4;
818      GOON,3;
819      ACTIVITY,,,RFP4;
820      ACTIVITY;
821      ACTIVITY,,,UUAC;
822      FREE,FORCREST,1;
823      ACTIVITY;
824      TERMINATE;
825  UUAC  FREE,CRANEFET,1;
826      ACTIVITY;
827      TERMINATE;
828  ;FILE P4FOTR.NET, NODE LABEL SEED MMMA
829  ;
830  RFP4  AWAIT(109),ALLOC(77),,1;
831      ACTIVITY/141,2880,,;REBFTP4;
832      GOON,3;
833      ACTIVITY;
834      ACTIVITY,,,CFP4;
835      ACTIVITY,,,MMMC;
836      FREE,CRANERBE,1;
837      ACTIVITY;
838      TERMINATE;
839  MMMC  FREE,RBCREST,1;
840      ACTIVITY;
841      TERMINATE;
842  ;FILE P4PEDC.NET, NODE LABEL SEED TPAA
843  ;
844  PCP4  EVENT,31,1;
845      ACTIVITY;
846  APP4  AWAIT(114),TRMIXRET,,1;
847      ACTIVITY/147,15,,;DELCONPEDP4;
848      FREE,TRMIXRET,1;
849      ACTIVITY;
850  CPP4  AWAIT(115),ALLOC(81),,1;
851      ACTIVITY/148,10,,;PLCONPEDP4;
852      FREE,CONPMPET,1;
853      ACTIVITY;
854      FREE,CONCREST,1;
855      ACTIVITY;
856      ACCUMULATE,80,80,,2;
857      ACTIVITY/149,480,,;RLPEDFRMP4;
858      ACTIVITY,,,SFP4;
859      FREE,PEDFRMET;
860      ACTIVITY;
861      TERMINATE;
862  ;FILE P4PEDF.NET, NODE LABEL SEED TUAA
863  ;
864  PFP4  AWAIT(112),ALLOC(79),,1;
```

```
865      ACTIVITY/145,1440,,;FRMPEDP4;
866      GOON,3;
867      ACTIVITY,,,PRP4;
868      ACTIVITY;
869      ACTIVITY,,,TUAD;
870      FREE,FORCREST,1;
871      ACTIVITY;
872      TERMINATE;
873  TUAD  FREE,CRANEFET,1;
874      ACTIVITY;
875      TERMINATE;
876  ;FILE P4PEDR.NET, NODE LABEL SEED TYAA
877  ;
878  PRP4  AWAIT(113),ALLOC(80),,1;
879      ACTIVITY/146,2880,,;REBPEDP4;
880      GOON,3;
881      ACTIVITY;
882      ACTIVITY,,,TYAC;
883      ACTIVITY,,,PCP4;
884      FREE,CRANERBE,1;
885      ACTIVITY;
886      TERMINATE;
887  TYAC  FREE,RBCREST,1;
888      ACTIVITY;
889      TERMINATE;
890  ;FILE P4SHC.NET, NODE LABEL SEED BXAA
891  ;
892  SCP4  EVENT,32,1;
893      ACTIVITY;
894  SP4A  AWAIT(118),TRMIXRET,,1;
895      ACTIVITY/152,15,,;DELCONSHP4;
896      FREE,TRMIXRET,1;
897      ACTIVITY;
898  SP42  AWAIT(119),ALLOC(84),,1;
899      ACTIVITY/153,10,,;PLCCONSHP4;
900      FREE,CONPMPET,1;
901      ACTIVITY;
902      FREE,CONCREST,1;
903      ACTIVITY;
904      ACCUMULATE,76,76,,2;
905      ACTIVITY/154,480,,;RELSHFRMP4;
906      ACTIVITY,,,BXAB;
907      FREE,SHFTFRET;
908      ACTIVITY;
909      TERMINATE;
910  BXAB  ASSIGN,ATRIB(10)=ATRIB(10) + 1,1;
911      ACTIVITY,,ATRIB(10).EQ.1,SFP4;
912      ACTIVITY,,ATRIB(10).EQ.2,FCP4;
913  ;FILE P4SHF.NET, NODE LABEL SEED LBAA
914  ;
915  SFP4  AWAIT(116),ALLOC(82),,1;
916      ACTIVITY/150,1440,,;FRMSHFTP4;
917      GOON,3;
918      ACTIVITY;
```

```
919      ACTIVITY,,,LBAC;
920      ACTIVITY,,,SRP4;
921      FREE,FORCREST,1;
922      ACTIVITY;
923      TERMINATE;
924 LBAC  FREE,CRANEFET,1;
925      ACTIVITY;
926      TERMINATE;
927 ;FILE P4SHR.NET, NODE LABEL SEED LUAA
928 ;
929 SRP4  AWAIT(117),ALLOC(83),,1;
930      ACTIVITY/151,2880,,;REBSHP4;
931      GOON,3;
932      ACTIVITY;
933      ACTIVITY,,,LUAC;
934      ACTIVITY,,,SCP4;
935      FREE,CRANERBE,1;
936      ACTIVITY;
937      TERMINATE;
938 LUAC  FREE,RBCREST,1;
939      ACTIVITY;
940      TERMINATE;
941 ;FILE P5BERM.NET, NODE LABEL SEED LLAA
942 ;
943 BRP5  EVENT,22,1;
944      ACTIVITY;
945 BP51  AWAIT(78),TRUCKEST,,1;
946      ACTIVITY/104,30,,;DELVDRTP5;
947      FREE,TRUCKEST,1;
948      ACTIVITY;
949 BP52  AWAIT(79),DOZEREST,,1;
950      ACTIVITY/105,20,,;DOZBP5;
951      FREE,DOZEREST;
952      ACTIVITY;
953      ACCUMULATE,800,800,,3;
954      ACTIVITY;
955      ACTIVITY,,,CDP5;
956      ACTIVITY,,,BRP4;
957      TERMINATE;
958 ;FILE P5BLIND.NET, NODE LABEL SEED ZZZA
959 ;
960 BLP5  EVENT,24,1;
961      ACTIVITY;
962 P5B1  AWAIT(88),TRMIXRET,,1;
963      ACTIVITY/117,15,,;DELCONCBP5;
964      FREE,TRMIXRET,1;
965      ACTIVITY;
966      AWAIT(89),ALLOC(62),,1;
967      ACTIVITY/118,10,,;PLACCONBP5;
968      FREE,CONPMPET,1;
969      ACTIVITY;
970      FREE,CONCREST,1;
971      ACTIVITY;
972      ACCUMULATE,5,5,,1;
```

```
973       ACTIVITY,,,FFP5;
974  ;FILE P5CAPC.NET, NODE LABEL SEED NYAA
975  ;
976  C1P5  EVENT,28,1;
977       ACTIVITY;
978  P5CA  AWAIT(104),TRMIXRET,,1;
979       ACTIVITY/136,15,,;DELCONCAPP5;
980       FREE,TRMIXRET,1;
981       ACTIVITY;
982  P54   AWAIT(105),ALLOC(74),,1;
983       ACTIVITY/137,10,,;PLCONCAPP5;
984       FREE,CONPMPET,1;
985       ACTIVITY;
986       FREE,CONCREST,1;
987       ACTIVITY;
988       ACCUMULATE,40,40,,2;
989       ACTIVITY/138,480,,;RELCAPFORMP5;
990       FREE,CAPFRMET;
991       TERMINATE;
992  ;FILE P5CAPF.NET, NODE LABEL SEED MOAA
993  ;
994  FCP5  AWAIT(102),ALLOC(72),,1;
995       ACTIVITY/134,1440,,;FORMCAPP5;
996       GOON,3;
997       ACTIVITY;
998       ACTIVITY,,,RCP5;
999       ACTIVITY,,,MOAC;
1000      FREE,FORCREST,1;
1001      ACTIVITY;
1002      TERMINATE;
1003 MOAC  FREE,CRANEFET,1;
1004      ACTIVITY;
1005      TERMINATE;
1006 ;FILE P5CAPR.NET, NODE LABEL SEED OMAA
1007 ;
1008 RCP5  AWAIT(103),ALLOC(73),,1;
1009      ACTIVITY/135,2880,,;REBCAPP5;
1010      GOON,3;
1011      ACTIVITY,,,C1P5;
1012      ACTIVITY;
1013      ACTIVITY,,,OMAD;
1014      FREE,CRANERBE,1;
1015      TERMINATE;
1016 OMAD  FREE,RBCREST,1;
1017      TERMINATE;
1018 ;FILE P5COFD.NET, NODE LABEL SEED KXYA
1019 ;
1020 CDP5  AWAIT(84),ALLOC(58),,1;
1021      ACTIVITY/111,240,,;FLOATCAGEP5;
1022      GOON,1;
1023      ACTIVITY/112,240,,;POS&ANCHP5;
1024 CP5A  AWAIT(85),ALLOC(59),,1;
1025      ACTIVITY,480,,;DRIVSPUDPILP5;
1026 KXYB  GOON,1;
```

```
1027      ACTIVITY/113,360,,;DRIV SHETP5;
1028      ASSIGN,ATRIB(7)=ATRIB(7) + 1,1;
1029      ACTIVITY,,ATRIB(7).GE.100;
1030      ACTIVITY,,ATRIB(7).LT.100,KXYB;
1031      FREE,CRANERBE,1;
1032      ACTIVITY;
1033      FREE,DISELHAM,1;
1034      ACTIVITY;
1035      FREE,PILCREW;
1036      ACTIVITY,,,BLP5;
1037  ;FILE P5FOTC.NET, NODE LABEL SEED MMMA
1038  ;
1039  CFP5  EVENT,25,1;
1040      ACTIVITY;
1041  P51   AWAIT(92),TRMIXRET,,1;
1042      ACTIVITY/121,15,,;DELCONFTP5;
1043      FREE,TRMIXRET,1;
1044      ACTIVITY;
1045  P52   AWAIT(93),ALLOC(65),,1;
1046      ACTIVITY/122,10,,;PLCCONP5;
1047      FREE,CONPMPET,1;
1048      ACTIVITY;
1049      FREE,CONCREST,1;
1050      ACTIVITY;
1051      ACCUMULATE,80,80,,2;
1052      ACTIVITY/123,480,,;RELFOTFRMP5;
1053      ACTIVITY,,,PFP5;
1054      FREE,FOTFORET;
1055      ACTIVITY;
1056      TERMINATE;
1057  ;FILE P5FOTF.NET, NODE LABEL SEED KKKA
1058  ;
1059  FFP5  AWAIT(90),ALLOC(63),,1;
1060      ACTIVITY/119,1440,,;FORMFTP5;
1061      GOON,3;
1062      ACTIVITY,,,RFP5;
1063      ACTIVITY;
1064      ACTIVITY,,,KKKC;
1065      FREE,FORCREST,1;
1066      ACTIVITY;
1067      TERMINATE;
1068  KKKC  FREE,CRANEFET,1;
1069      ACTIVITY;
1070      TERMINATE;
1071  ;FILE P5FOTR.NET, NODE LABEL SEED LLLA
1072  ;
1073  RFP5  AWAIT(91),ALLOC(64),,1;
1074      ACTIVITY/120,2880,,;REBFTP5;
1075      GOON,3;
1076      ACTIVITY;
1077      ACTIVITY,,,CFP5;
1078      ACTIVITY,,,LLLC;
1079      FREE,CRANERBE,1;
1080      ACTIVITY;
```

```
1081      TERMINATE;
1082 LLLC  FREE,RBCREST,1;
1083      ACTIVITY;
1084      TERMINATE;
1085 ;FILE P5PEDC.NET, NODE LABEL SEED NTAA
1086 ;
1087 PCP5  EVENT,26,1;
1088      ACTIVITY;
1089 APP5  AWAIT(96),TRMIXRET,,1;
1090      ACTIVITY/126,15,,;DELCONPEDP5;
1091      FREE,TRMIXRET,1;
1092      ACTIVITY;
1093 CPP5  AWAIT(97),ALLOC(68),,1;
1094      ACTIVITY/127,10,,;PLCONPEDP5;
1095      FREE,CONPMPET,1;
1096      ACTIVITY;
1097      FREE,CONCREST,1;
1098      ACTIVITY;
1099      ACCUMULATE,80,80,,2;
1100      ACTIVITY/128,480,,;RLPEDFRMP5;
1101      ACTIVITY,,,SFP5;
1102      FREE,PEDFRMET;
1103      ACTIVITY;
1104      TERMINATE;
1105 ;FILE P5PEDF.NET, NODE LABEL SEED IIAA
1106 ;
1107 PFP5  AWAIT(94),ALLOC(66),,1;
1108      ACTIVITY/124,1440,,;FRMPEDP5;
1109      GOON,3;
1110      ACTIVITY,,,PRP5;
1111      ACTIVITY;
1112      ACTIVITY,,,IIAB;
1113      FREE,FORCREST,1;
1114      ACTIVITY;
1115      TERMINATE;
1116 IIAB  FREE,CRANEFET,1;
1117      ACTIVITY;
1118      TERMINATE;
1119 ;FILE P5PEDR.NET, NODE LABEL SEED LTAA
1120 ;
1121 PRP5  AWAIT(95),ALLOC(67),,1;
1122      ACTIVITY/125,2880,,;REBPEDP5;
1123      GOON,3;
1124      ACTIVITY;
1125      ACTIVITY,,,LTAC;
1126      ACTIVITY,,,PCP5;
1127      FREE,CRANERBE,1;
1128      ACTIVITY;
1129      TERMINATE;
1130 LTAC  FREE,RBCREST,1;
1131      ACTIVITY;
1132      TERMINATE;
1133 ;FILE P5SHC.NET, NODE LABEL SEED ITAA
1134 ;
```

```
SCP5  EVENT,27,1;
      ACTIVITY;
SPP1  AWAIT(100),TRMIXRET,,1;
      ACTIVITY/131,15,,;DELCONSHP1;
      FREE,TRMIXRET,1;
      ACTIVITY;
SPP2  AWAIT(101),ALLOC(71),,1;
      ACTIVITY/132,10,,;PLCCONSHP5;
      FREE,CONPMPET,1;
      ACTIVITY;
      FREE,CONCREST,1;
      ACTIVITY;
      ACCUMULATE,76,76,,2;
      ACTIVITY/133,480,,;RELSHFRMP5;
      ACTIVITY,,,ITAB;
      FREE,SHFTFRET;
      ACTIVITY;
      TERMINATE;
ITAB  ASSIGN,ATRIB(9)=ATRIB(9) + 1,1;
      ACTIVITY,,ATRIB(9).EQ.1,SFP5;
      ACTIVITY,,ATRIB(9).EQ.2,FCP5;
;FILE P5SHF.NET, NODE LABEL SEED TIAA
;
SFP5  AWAIT(98),ALLOC(69),,1;
      ACTIVITY/129,1440,,;FRMSHFTP5;
      GOON,3;
      ACTIVITY;
      ACTIVITY,,,TIAC;
      ACTIVITY,,,SRP5;
      FREE,FORCREST,1;
      ACTIVITY;
      TERMINATE;
TIAC  FREE,CRANEFET,1;
      ACTIVITY;
      TERMINATE;
;FILE P5SHR.NET, NODE LABEL SEED MEAA
;
SRP5  AWAIT(99),ALLOC(70),,1;
      ACTIVITY/130,2880,,;REBSHP5;
      GOON,3;
      ACTIVITY;
      ACTIVITY,,,MEAC;
      ACTIVITY,,,SCP5;
      FREE,CRANERBE,1;
      ACTIVITY;
      TERMINATE;
MEAC  FREE,RBCREST,1;
      ACTIVITY;
      TERMINATE;
;FILE P6BLIND.NET, NODE LABEL SEED ZZZZ
;
BLP6  EVENT,17,1;
      ACTIVITY;
P6B1  AWAIT(60),TRMIXRET,,1;
```

```
1189      ACTIVITY/82,15,,;DELCONCBP6;
1190      FREE,TRMIXRET,1;
1191      ACTIVITY;
1192      AWAIT(61),ALLOC(42),,1;
1193      ACTIVITY/83,10,,;PLACCONBP6;
1194      FREE,CONPMPET,1;
1195      ACTIVITY;
1196      FREE,CONCREST,1;
1197      ACTIVITY;
1198      ACCUMULATE,5,5,,1;
1199      ACTIVITY,,,FFP6;
1200  ;FILE P6CAPC.NET, NODE LABEL SEED KNAA
1201  ;
1202  C1P6  EVENT,21,1;
1203      ACTIVITY;
1204  P6CA  AWAIT(76),TRMIXRET,,1;
1205      ACTIVITY/101,15,,;DELCONCAPP6;
1206      FREE,TRMIXRET,1;
1207      ACTIVITY;
1208  P64   AWAIT(77),ALLOC(55),,1;
1209      ACTIVITY/102,10,,;PLCONCAPP6;
1210      FREE,CONPMPET,1;
1211      ACTIVITY;
1212      FREE,CONCREST,1;
1213      ACTIVITY;
1214      ACCUMULATE,40,40,,2;
1215      ACTIVITY/103,480,,;RELCAPFORMP6;
1216      FREE,CAPFRMET;
1217      TERMINATE;
1218  ;FILE P6CAPF.NET, NODE LABEL SEED KMAA
1219  ;
1220  FCP6  AWAIT(74),ALLOC(53),,1;
1221      ACTIVITY/99,1440,,;FORMCAPP6;
1222      GOON,3;
1223      ACTIVITY;
1224      ACTIVITY,,,RCP6;
1225      ACTIVITY,,,KMAB;
1226      FREE,FORCREST,1;
1227      ACTIVITY;
1228      TERMINATE;
1229  KMAB  FREE,CRANEFET,1;
1230      ACTIVITY;
1231      TERMINATE;
1232  ;FILE P6CAPR.NET, NODE LABEL SEED LKAA
1233  ;
1234  RCP6  AWAIT(75),ALLOC(54),,1;
1235      ACTIVITY/100,2880,,;REBCAPP6;
1236      GOON,3;
1237      ACTIVITY,,,C1P6;
1238      ACTIVITY;
1239      ACTIVITY,,,LKAB;
1240      FREE,CRANERBE,1;
1241      TERMINATE;
1242  LKAB  FREE,RBCREST,1;
```

```
1243      TERMINATE;
1244  ;FILE P6FOTC.NET, NODE LABEL SEED ZXAA
1245  ;
1246  CFP6  EVENT,18,1;
1247      ACTIVITY;
1248  P61   AWAIT(64),TRMIXRET,,1;
1249      ACTIVITY/86,15,,;DELCONFTP6;
1250      FREE,TRMIXRET,1;
1251      ACTIVITY;
1252  P62   AWAIT(65),ALLOC(46),,1;
1253      ACTIVITY/87,10;
1254      FREE,CONPMPET,1;
1255      ACTIVITY;
1256      FREE,CONCREST,1;
1257      ACTIVITY;
1258      ACCUMULATE,80,80,,2;
1259      ACTIVITY/88,480,,;RELFOTFRMP6;
1260      ACTIVITY,,,PFP6;
1261      FREE,FOTFORET;
1262      ACTIVITY;
1263      TERMINATE;
1264  ;FILE P6FOTF.NET, NODE LABEL SEED ZZZB
1265  ;
1266  FFP6  AWAIT(62),ALLOC(43),,1;
1267      ACTIVITY/84,1440,,;FORMFTP6;
1268      GOON,3;
1269      ACTIVITY,,,RFP6;
1270      ACTIVITY;
1271      ACTIVITY,,,ZZZC;
1272      FREE,FORCREST,1;
1273      ACTIVITY;
1274      TERMINATE;
1275  ZZZC  FREE,CRANEFET,1;
1276      ACTIVITY;
1277      TERMINATE;
1278  ;FILE P6FOTR.NET, NODE LABEL SEED ZZAA
1279  ;
1280  RFP6  AWAIT(63),ALLOC(44),,1;
1281      ACTIVITY/85,2880,,;REBFTP6;
1282      GOON,3;
1283      ACTIVITY;
1284      ACTIVITY,,,CFP6;
1285      ACTIVITY,,,ZZAC;
1286      FREE,CRANERBE,1;
1287      ACTIVITY;
1288      TERMINATE;
1289  ZZAC  FREE,RBCREST,1;
1290      ACTIVITY;
1291      TERMINATE;
1292  ;FILE P6PEDC.NET, NODE LABEL SEED XXAA
1293  ;
1294  PCP6  EVENT,19,1;
1295      ACTIVITY;
1296  APP6  AWAIT(68),TRMIXRET,,1;
```

```
1297      ACTIVITY/91,15,,;DELCONPEDP6;
1298      FREE,TRMIXRET,1;
1299      ACTIVITY;
1300 CPP6  AWAIT(69),ALLOC(49),,1;
1301      ACTIVITY/92,10,,;PLCONPEDP6;
1302      FREE,CONPMPET,1;
1303      ACTIVITY;
1304      FREE,CONCREST,1;
1305      ACTIVITY;
1306      ACCUMULATE,80,80,,2;
1307      ACTIVITY/93,480,,;RLPEDFRMP6;
1308      ACTIVITY,,,SFP6;
1309      FREE,PEDFRMET;
1310      ACTIVITY;
1311      TERMINATE;
1312 ;FILE P6PEDF.NET, NODE LABEL SEED ZYAA
1313 ;
1314 PFP6  AWAIT(66),ALLOC(47),,1;
1315      ACTIVITY/89,1440,,;FRMPEDP6;
1316      GOON,3;
1317      ACTIVITY,,,PRP6;
1318      ACTIVITY;
1319      ACTIVITY,,,ZYAB;
1320      FREE,FORCREST,1;
1321      ACTIVITY;
1322      TERMINATE;
1323 ZYAB  FREE,CRANEFET,1;
1324      ACTIVITY;
1325      TERMINATE;
1326 ;FILE P6PEDR.NET, NODE LABEL SEED ZTAA
1327 ;
1328 PRP6  AWAIT(67),ALLOC(48),,1;
1329      ACTIVITY/90,2880,,;REBPEDP6;
1330      GOON,3;
1331      ACTIVITY;
1332      ACTIVITY,,,ZTAB;
1333      ACTIVITY,,,PCP6;
1334      FREE,CRANERBE,1;
1335      ACTIVITY;
1336      TERMINATE;
1337 ZTAB  FREE,RBCREST,1;
1338      ACTIVITY;
1339      TERMINATE;
1340 ;FILE P6SHC.NET, NODE LABEL SEED KKAA
1341 ;
1342 SCP6  EVENT,20,1;
1343      ACTIVITY;
1344 SCP1  AWAIT(72),TRMIXRET,,1;
1345      ACTIVITY/96,15,,;DELCONSHP6;
1346      FREE,TRMIXRET,1;
1347      ACTIVITY;
1348 SCP2  AWAIT(73),ALLOC(52),,1;
1349      ACTIVITY/97,10,,;PLCCONSHP6;
1350      FREE,CONPMPET,1;
```

```
1351      ACTIVITY;
1352      FREE,CONCREST,1;
1353      ACTIVITY;
1354      ACCUMULATE,76,76,,2;
1355      ACTIVITY/98,480,,;RELSHFRMP6;
1356      ACTIVITY,,,KKAB;
1357      FREE,SHFTFRET;
1358      ACTIVITY;
1359      TERMINATE;
1360 KKAB  ASSIGN,ATRIB(6)=ATRIB(6) + 1,1;
1361      ACTIVITY,,ATRIB(6).EQ.1,SFP6;
1362      ACTIVITY,,ATRIB(6).EQ.2,FCP6;
1363 ;FILE P6SHF.NET, NODE LABEL SEED XYAA
1364 ;
1365 SFP6  AWAIT(70),ALLOC(50),,1;
1366      ACTIVITY/94,1440,,;FRMSHFTP6;
1367      GOON,3;
1368      ACTIVITY;
1369      ACTIVITY,,,XYAB;
1370      ACTIVITY,,,SRP6;
1371      FREE,FORCREST,1;
1372      ACTIVITY;
1373      TERMINATE;
1374 XYAB  FREE,CRANEFET,1;
1375      ACTIVITY;
1376      TERMINATE;
1377 ;FILE P6SHR.NET, NODE LABEL SEED XZAA
1378 ;
1379 SRP6  AWAIT(71),ALLOC(51),,1;
1380      ACTIVITY/95,2880,,;REBSHP6;
1381      GOON,3;
1382      ACTIVITY;
1383      ACTIVITY,,,XZAB;
1384      ACTIVITY,,,SCP6;
1385      FREE,CRANERBE,1;
1386      ACTIVITY;
1387      TERMINATE;
1388 XZAB  FREE,RBCREST,1;
1389      ACTIVITY;
1390      TERMINATE;
1391 ;FILE PILAB1.NET, NODE LABEL SEED ZAAA
1392 ;
1393 SHET  CREATE,0,,,32,1;
1394      ACTIVITY;
1395 PAB1  AWAIT(1),ALLOC(1),,1;
1396      ACTIVITY/1,10,,;POSITION PIL
1397 AG1   GOON,1;
1398      ACTIVITY/2,30,,;DRIVE-1;
1399 AG2   GOON,1;
1400      ACTIVITY/3,20,,;SPLICE;
1401 AG3   GOON,1;
1402      ACTIVITY/4,30,,;DRIVE-2;
1403 AG4   GOON,3;
1404      ACTIVITY;
```

```
1405      ACTIVITY,,,FCR;
1406      ACTIVITY,,,FHAM;
1407 FCRN  FREE,CRANE40T,1;
1408      ACTIVITY;
1409      TERMINATE;
1410 FCR   FREE,PILCREW,1;
1411      ACTIVITY;
1412      TERMINATE;
1413 FHAM  FREE,DISELHAM,1;
1414      ACTIVITY;
1415      TERMINATE;
1416 ;FILE PILPR1.NET, NODE LABEL SEED ZAAA
1417 ;
1418 PLP1  EVENT,1,1;
1419      ACTIVITY;
1420 PLPR  AWAIT(4),ALLOC(3),,1;
1421      ACTIVITY/8,10,,;POSITION PIL;
1422 PG1   GOON,1;
1423      ACTIVITY/9,30,,;DRIVE-1;
1424 PG2   GOON,1;
1425      ACTIVITY/10,20,,;SPLICE;
1426 PG3   GOON,1;
1427      ACTIVITY/11,30,,;DRIVE-2;
1428 PG4   GOON,4;
1429      ACTIVITY;
1430      ACTIVITY,,,ZAAC;
1431      ACTIVITY,,,XAAB;
1432      ACTIVITY,,,XAAC;
1433 BREL  ACCUMULATE,90,90,,1;
1434      ACTIVITY,,,BLP1;
1435 ZAAC  FREE,CRANE40T,1;
1436      ACTIVITY;
1437      TERMINATE;
1438 XAAB  FREE,PILCREW,1;
1439      ACTIVITY;
1440      TERMINATE;
1441 XAAC  FREE,DISELHAM,1;
1442      ACTIVITY;
1443      TERMINATE;
1444 ;FILE REBFTP1.NET, NODE LABEL SEED TAAA
1445 ;
1446 RFP1  AWAIT(8),ALLOC(6),,1;
1447      ACTIVITY/15,2880,,;REBFTP1;
1448      GOON,3;
1449      ACTIVITY;
1450      ACTIVITY,,,TAAB;
1451      ACTIVITY,,,CFP1;
1452      FREE,CRANERB,1;
1453      TERMINATE;
1454 TAAB  FREE,RBCRWST,1;
1455      TERMINATE;
1456 ;FILE REBSHP1.NET, NODE LABEL SEED QAAA
1457 ;
1458 RFSH  AWAIT(12),ALLOC(9),,1;
```

```
1459      ACTIVITY/20,2880,,:REBSHP1;
1460      GOON,3;
1461      ACTIVITY;
1462      ACTIVITY,,,QAAB;
1463      ACTIVITY,,,COSH;
1464      FREE,CRANERB,1;
1465      TERMINATE;
1466 QAAB  FREE,RBCRWST,1;
1467      TERMINATE;
1468 ;FILE SHCONP1.NET, NODE LABEL SEED PAAA
1469 ;
1470 COSH  EVENT,4,1;
1471      ACTIVITY;
1472      AWAIT(13),TRMIXRWT,,1;
1473      ACTIVITY/21,15,,:DEL CON SH P1;
1474      FREE,TRMIXRWT,1;
1475      ACTIVITY;
1476      AWAIT(14),ALLOC(10),,1;
1477      ACTIVITY/22,10,,:PLACECONCSHP1;
1478      FREE,CONPMPWT,1;
1479      ACTIVITY;
1480      FREE,CONCRWST,1;
1481      ACTIVITY;
1482      ACCUMULATE,76,76,,2;
1483      ACTIVITY/23,480,,:REL SHTFORM;
1484      ACTIVITY,,,PAAB;
1485      FREE,SHFTFRWT;
1486      TERMINATE;
1487 PAAB  ASSIGN,ATRIB(2)=ATRIB(2) +1,1;
1488      ACTIVITY,,ATRIB(2).EQ.1,SHP1;
1489      ACTIVITY,,ATRIB(2).EQ.2,CAPF;
1490 ;FILE SHFORP1.NET, NODE LABEL SEED RAAA
1491 ;
1492 SHP1  AWAIT(11),ALLOC(8),,1;
1493      ACTIVITY/19,1440,,:FRMSHFTP1;
1494      GOON,3;
1495      ACTIVITY;
1496      ACTIVITY,,,RAAB;
1497      ACTIVITY,,,RFSH;
1498      FREE,FORCRWST,1;
1499      TERMINATE;
1500 RAAB  FREE,CRANEFWT,1;
1501      TERMINATE;
1502 ;FILE START.NET, NODE LABEL SEED ACAA
1503 ;
1504      CREATE,,,,1;
1505      ACTIVITY,,,BLP6;
1506      ACTIVITY,,,BRP5;
1507      END;
1508 FIN;
```